```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
```

```
XX      XX  DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
XX      XX  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    XX      DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    XX      DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
  XX  XX    DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
  XX  XX    DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
XX      XX  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  DDDDDDDD    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
XX      XX  DDDDDDDD    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

(2) 115 External and local symbol definitions
(9) 685 Standard tables
(10) 781 P2 buffer verification tables
(11) 1100 UNIT_INIT, Unit initialization routine
(12) 1208 XMT_FDT, Transmit I/O Operation FDT Routine
(13) 1309 XMT_START, Start Transmit Operation
(14) 1483 RCV_FDT, Read I/O Operation FDT Routine
(15) 1554 RCV_START, Start Receive Operation
(16) 1625 ALT_START, Alternate Start I/O Routine
(17) 1703 SETMODE_FDT, Set mode I/O operation FDT routine
(18) 1994 SETMODE_CTRL, Perform setmode FDT operation on controller
(19) 2142 GET_CHAR_BUFS, Get P1 and P2 characteristics buffers
(20) 2210 SENSEMODE_FDT, Sense Mode I/O operation FDT routine
(21) 2354 SENSEMODE_CTRL, Perform SENSEMODE FDT processing for controller
(22) 2452 SENSE_MODEM, Perform SENSEMODE READ_MODEM FDT processing
(23) 2501 CHECK_BUFS, Check P1 and P2 buffers for write access
(24) 2554 CHECK_P1, Check P1 buffer address for write access
(25) 2595 ALLOC_P2BUF, Allocate a P2 buffer and charge user's quota
(26) 2664 STARTIO, Start I/O routine
(27) 2747 STARTUP, Device initialization routine
(28) 2932 START_CONT, Continue after timeout
(29) 3011 START_TRIB, Start tributary routine
(30) 3047 DEV_ACTIVE, Device must be active
(31) 3083 TRIB_ACTIVE, Tributary must be active
(32) 3119 TRIB_ERRS, Read trib errors routine
(33) 3176 FILLRCVLIST, Fill receive buffer list
(33) 3177 ADDRCVLIST, Add a buffer to the receive list
(34) 3246 START_RECEIVE, Start any receive requests pending
(35) 3322 LOAD_PORT, Request CSR routine
(36) 3914 RDI_INTRPT, CSR Ready Input Interrupt Service Routine
(37) 3981 RDO_INTRPT, CSR Ready Output Interrupt Service Routine
(38) 4296 OUT_STATUS, Process Information Out from device
(39) 4414 SCHED_FORK, Schedule the fork process
(39) 4415 SCHED_FORKC, Schedule the fork process with R3 clear
(40) 4465 FORK_PROC, Error and completion fork process handling
(41) 4759 FINISH_RCV_IO, Finish receive I/O processing
(42) 4838 HALT_TRIBS, Halt all tribs that were forced to shutdown
(43) 4870 START_XMITS, Start xmits for tribs in run state
(44) 4902 SHUTDOWN_TRIB, Shutdown the tributary
(45) 5026 SHUTDOWN_DEV, Shutdown the controller
(46) 5146 REG_DUMP, Device register dump routine
(47) 5178 RETURN_IRP, Deallocate IRP
(47) 5179 RETURN_CDB, Deallocate CDB
(47) 5180 CLEAR_CDB, Deallocate CDB
(48) 5233 FIND_SLOT, Find an empty slot in translation vector
(49) 5279 FIND_TRIB, Find CDB from trib address
(50) 5319 XLATE, Translate PID and Channel to CDB address
(50) 5320 XLATE_ALT, Translate PID and Channel to CDB address
(51) 5411 GET_CDB, Get CDB address from IRP
(52) 5443 CHG_UCB, Change UCB parameter values
(53) 5561 CHG_CDB, Change CDB parameter values
(54) 5619 VALIDATE_P2, Validate P2 buffer parameters
(54) 5620 VALIDATE_P2_CDB, Validate P2 buffer with CDB
(54) 5621 VALIDATE_P2_UCB, Validate P2 buffer with UCB
(55) 5764 UPDATE_P2, Update UCB/CDB based on P2 buffer parameters
(56) 5839 RETURN_P2, Return UCB/CDB buffer parameters
(57) 5896 UNPACK_P2_BUF, Unpack a P2 parameter from P2 buffer
(58) 5941 POKE_USER, Deliver attention AST
(59) 5986 CANCEL, Cancel I/O routine
(60) 6180 INIT_UCB, Initialize UCB parameters

(61) 6219 Timer setup routines
(62) 6277 TIMEOUT, Timecut Routine
(63) 6323 XD_FILL_JNX, Fill a journal record
(63) 6324 XD_BUF_JNX, Get a journal record

```
0000     1          .NOSHOW  CONDITIONALS
0000     2
0000     3          .TITLE  XDDRIVER - VAX/VMS DMP11/DMV11 Device Driver
0000     4          .IDENT  'V04-000'
0000     5
0000     6  ;******************************************************************************
0000     7  ;*                                                                            *
0000     8  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                   *
0000     9  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
0000    10  ;*  ALL RIGHTS RESERVED.                                                      *
0000    11  ;*                                                                            *
0000    12  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED      *
0000    13  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE     *
0000    14  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER      *
0000    15  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY      *
0000    16  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY      *
0000    17  ;*  TRANSFERRED.                                                               *
0000    18  ;*                                                                            *
0000    19  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE      *
0000    20  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
0000    21  ;*  CORPORATION.                                                               *
0000    22  ;*                                                                            *
0000    23  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    24  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
0000    25  ;*                                                                            *
0000    26  ;*                                                                            *
0000    27  ;******************************************************************************
0000    28  ;++
0000    29  ; FACILITY:
0000    30  ;
0000    31  ;       VAX/VMS DMP11/DMV11 Device driver
0000    32  ;
0000    33  ; ABSTRACT:
0000    34  ;
0000    35  ;       This module contains the DMP11/DMV11 driver FDT routines,
0000    36  ;       interrupt dispatcher, interrupt service and fork routines.
0000    37  ;
0000    38  ; AUTHOR:
0000    39  ;
0000    40  ;       R. GAMACHE      13-JUL-81
0000    41  ;
0000    42  ; MODIFICATION HISTORY:
0000    43  ;
0000    44  ;       V03-036 RNG0036         Rod N. Gamache          7-Aug-1984
0000    45  ;               Make sure requests on the XMIT Pending queue are not aborted
0000    46  ;               without releasing map registers, when a trib is shutdown.
0000    47  ;               Set dead threshold from 16 to 3.
0000    48  ;
0000    49  ;       V03-035 RNG0035         Rod N. Gamache          4-May-1984
0000    50  ;               Set DEV$M_NET and DEV$M_AVL bits in device characteristics.
0000    51  ;
0000    52  ;       V03-034 RNG0034         Rod N. Gamache         19-Apr-1984
0000    53  ;               Don't allow transmits larger than 16 KB.
0000    54  ;
0000    55  ;       V03-033 RNG0033         Rod N. Gamache         14-Dec-1983
0000    56  ;               Fix max tribs for DMV11 to be 16 instead of 32.
0000    57  ;               Fix problem with "streaming tribs" and losing the
```

```
0000     58 ;                transmit slots.
0000     59 ;                Get descriptor size as a word rather than a longword.
0000     60 ;
0000     61 ;        V03-032 RNG0032         Rod N. Gamache          13-Sep-1983
0000     62 ;                Add support for the DMV11 in 22-bit mode.
0000     63 ;
0000     64 ;        V03-031 RNG0031         Rod N. Gamache          31-Aug-1983
0000     65 ;                Make all NOOP function request clear BSEL7.
0000     66 ;                Aborted INPUT CANCEL requests now release the input CSRs.
0000     67 ;                CANCEL no longer aborts transmits in progress prematurely,
0000     68 ;                this allows the FORK process to return map registers.
0000     69 ;
0000     70 ;        V03-030 RNG0030         Rod N. Gamache          29-Jun-1983
0000     71 ;                Fix the XMIT and RECV operations to check the ACTIVE
0000     72 ;                bit rather than the ESTAB bit when performing I/O operations.
0000     73 ;                Set up the P1 characteristics on a SENSEMODE after finding
0000     74 ;                the polling state.
0000     75 ;                Add journaling to file.
0000     76 ;
0000     77 ;        V03-029 RNG0029         Rod N. Gamache          23-Jun-1983
0000     78 ;                Remove internal definition of IRP$Q_STATION (for V4 only)
0000     79 ;
0000     80 ;        V03-028 RNG0028         Rod N. Gamache          12-May-1983
0000     81 ;                Lots of cleanup for MOP mode.
0000     82 ;
0000     83 ;        V03-027 RNG0027         Rod N. Gamache          09-May-1983
0000     84 ;                Wait a little while before issueing a STOP tributary
0000     85 ;                request. This is in case we just processed a $CANCEL
0000     86 ;                request and the trib hasn't finished running down.
0000     87 ;
0000     88 ;        V03-026 RNG0025         Rod N. Gamache          20-Jan-1983
0000     89 ;                Initialize the UCB Fork Process address to be the fork
0000     90 ;                Process routine and not the timeout routine. This is
0000     91 ;                because in the event a of a device timeout, before the
0000     92 ;                device is really initialized, then the timeout routine
0000     93 ;                does not need to have an offset to the timeout routine.
0000     94 ;                Don't call GET_CDB on HALTT request!
0000     95 ;                Change default Device IPL to be 21 (instead of 22).
0000     96 ;
0000     97 ;        V03-025 RNG0024         Rod N. Gamache          3-Dec-1982
0000     98 ;                Don't reset the tributary DEVDEPEND field on HALTT request.
0000     99 ;
0000    100 ;        V03-024 RNG0023         Rod N. Gamache          4-Nov-1982
0000    101 ;                Save R4 in Write FDT routine. Setup address of timeout
0000    102 ;                routine for new fork process in transmit routine.
0000    103 ;
0000    104 ;        V03-023 RNG0022         Rod N. Gamache          21-Sep-1982
0000    105 ;                Add another fork block to UCB to allow a fork on transmit
0000    106 ;                requests - allows users to transmit any size message up to
0000    107 ;                16K. Remove the code to pre-allocate one transmit map register.
0000    108 ;
0000    109 ;        V03-022 RNG0021         Rod N. Gamache          27-Aug-1982
0000    110 ;                Fix the CANCEL routine, to only abort I/O on $CANCEL
0000    111 ;                request and not halt the tributary.
0000    112 ;
0000    113 ;--
```

```
                        .SBTTL  External and local symbol definitions

;
; System definitions
;

        $ACBDEF                         ; AST control block
        $ADPDEF                         ; Adapter control block
        $CANDEF                         ; Define $CANCEL reason codes
        $CRBDEF                         ; Channel request block
        $CXBDEF                         ; Define CXB header
        $DCDEF                          ; Device classes and types
        $DDBDEF                         ; Device data block
        $DEVDEF                         ; Device characteristics
        $DPTDEF                         ; Driver prologue table defs
        $DYNDEF                         ; Control block defs
        $FKBDEF                         ; Fork block definitions
        $IDBDEF                         ; Interrupt data block
        $IODEF                          ; I/O function codes
        $IPLDEF                         ; Hardware IPL definitions
        $IRPDEF                         ; I/O request packet
        $JIBDEF                         ; Job information block defs
        $NMADEF                         ; Network Management definitions
        $PCBDEF                         ; Process control block
        $PRDEF                          ; Processor register definitions
        $PTEDEF                         ; Define system PTEs
        $SSDEF                          ; System status codes
        $UBADEF                         ; UBA definitions
        $UCBDEF                         ; Unit control block
        $VADEF                          ; Virtual address defs
        $VECDEF                         ; Interrupt vector block
        $XMDEF                          ; XMDRIVER symbols

;
; Local symbol definitions
;

00000001    JNX$$$ = 1                  ; Enables journaling

;
; $QIO parameter offsets
;

00000000    P1      = 0                 ; Parameter 1
00000004    P2      = 4                 ; Parameter 2
00000008    P3      = 8                 ; Parameter 3
0000000C    P4      = 12                ; Parameter 4
00000010    P5      = 16                ; Parameter 5

;
; Constants
;

00000007    MAX_RCV = 7                 ; Maximum number outstanding receives
00000007    MAX_XMT = 7                 ; Maximum number outstanding transmits
00000004    MAX_RCV_UV1 = 4             ; Maximum number of receives on u-VAX I
00000002    MAX_XMT_UV1 = 2             ; Maximum number of transmits on u-VAX I
00003FFF    MAX_BUFSIZ  = 16383         ; Maximum DMP/DMV buffer size
```

```
00000400  0000   172 MAX_BUFSIZ_UV1 = 1024                          ; Maximum size buffer on u-VAX I
00001800  0000   173 UV1_BUFFER_AREA = <MAX_XMT_UV1*MAX_BUFSIZ_UV1>+- ; Size of u-VAX physically
          0000   174                   <MAX_RCV_UV1*MAX_BUFSIZ_UV1> ;  contiguous buffer area
000000FF  0000   175 INF_RCV = 255                                 ; "Logical" infinite receives (unlimited)
0000007F  0000   176 INF_XMT = 127                                 ; "Logical" infinite transmits
          0000   177
00000200  0000   178 XD_DEF_BUFSIZ   = 512                         ; Default buffer size
00000020  0000   179 MAX_DMP_TRIB    = 32                          ; Max tributaries on DMP11 device
00000010  0000   180 MAX_DMV_TRIB    = 16                          ; Max tributaries on DMV11 device
00000004  0000   181 MAX_XMT_TRB     = 4                           ; Maximum number of xmits per trib
00000004  0000   182 DSC$A_POINTER   = 4                           ; Descriptor buffer address field
          0000   183 P1_TRIB_MASK    = XM$M_STS_ACTIVE!-           ; Mask of user "read only" bits
000028FF  0000   184                   XM$M_STS_RUNNING!^X<FF>     ; in CDB device dependent longword
000000FF  0000   185 P1_LINE_MASK    = ^X<FF>                      ; Mask of user "read only" bits
          0000   186
          0000   187 ;
          0000   188 ; Journal buffer definitions
          0000   189 ;
00000001  0000   190 JNL_XMT = 1                                   ; XMIT operation
00000002  0000   191 JNL_RCV = 2                                   ; RECV operation
00000003  0000   192 JNL_QIO = 3                                   ; QIO (SETMODE/SENSEMODE) operation
00000004  0000   193 JNL_QUE = 4                                   ; QUEUED PACKET operation
00000005  0000   194 JNL_POST = 5                                  ; POSTED PACKET operation
00000006  0000   195 JNL_XMTA = 6                                  ; Altstart XMIT operation
00000007  0000   196 JNL_RCVA = 7                                  ; Altstart RECV operation
00000008  0000   197 JNL_CSRIN = 8                                 ; Input to CSRs (same as QUE)
00000009  0000   198 JNL_CSROUT = 9                                ; Output CSRs
0000000A  0000   199 JNL_CANC = 10                                 ; Cancel request
```

```
0000   201                                          ; in UCB device dependent longword
0000   202 ;
0000   203 ; Local macros
0000   204 ;
0000   205
0000   206          .MACRO  SETBIT  POS,BAS,?L       ; Set a single bit
0000   207                  BBSS    POS,BAS,L
0000   208          L:
0000   209          .ENDM   SETBIT
0000   210
0000   211          .MACRO  CLRBIT  POS,BAS,?L       ; Clear a single bit
0000   212                  BBCC    POS,BAS,L
0000   213          L:
0000   214          .ENDM   CLRBIT
0000   215
0000   216          .MACRO  INCC    COUNTER,CONTEXT=L,?L  ; Increment counter
0000   217                  INC'CONTEXT     COUNTER ; Do Increment
0000   218                  BCC     L               ; Br if no carry set
0000   219                  DEC'CONTEXT     COUNTER ; Leave at maximum value
0000   220          L:
0000   221          .ENDM   INCC
0000   222
0000   223          .MACRO  PUSHQ   ARG             ; Push a quadword
0000   224                  MOVQ    ARG,-(SP)       ; Save argument on stack
0000   225          .ENDM   PUSHQ
0000   226
0000   227          .MACRO  POPQ    ARG             ; Pop a quadword
0000   228                  MOVQ    (SP)+,ARG       ; Restore argument
0000   229          .ENDM   POPQ
0000   230
0000   231          .MACRO  PARAM   TYPE,OFFSET,WIDTH,MIN,MAX,INVALID,BASE
0000   232 ;
0000   233 ; Inputs:
0000   234 ;
0000   235 ;        TYPE = Parameter type
0000   236 ;        OFFSET = Offset in UCB/CDB to current value
0000   237 ;        WIDTH = Width of field in UCB/CDB (B,W,L)
0000   238 ;        MIN = Minimum value parameter is allowed to take
0000   239 ;        MAX = Maximum value parameter is allowed to take
0000   240 ;        INVALID = Invalid flags in status word
0000   241 ;        BASE = Data base (LINE,TRIB)
0000   242 ;
0000   243          .IF BLANK TYPE
0000   244          .WORD   0
0000   245          .IF FALSE
0000   246          $$$TYP = TYPE & PRM_M_TYPE              ; Isolate type code
0000   247          .IIF NOT_BLANK  <MIN>, $$$TYP = $$$TYP!PRM_M_MIN
0000   248          .IIF NOT_BLANK  <MAX>, $$$TYP = $$$TYP!PRM_M_MAX
0000   249          .IIF NOT_BLANK  <INVALID>, $$$TYP = $$$TYP!PRM_M_INVALID
0000   250                  .WORD   $$$TYP
0000   251          $$$OFF = OFFSET & OFF_M_VALUE           ; Isolate offset only
0000   252          $$$WID = 0                              ; Set null width
0000   253          .IIF IDN <WIDTH><B>, $$$WID = <1@OFF_V_WIDTH>
0000   254          .IIF IDN <WIDTH><W>, $$$WID = <2@OFF_V_WIDTH>
0000   255          .IIF IDN <WIDTH><L>, $$$WID = <3@OFF_V_WIDTH>
0000   256                  .WORD   $$$OFF!$$$WID
0000   257          .IIF NOT_BLANK  <MIN>, .WORD    MIN
```

```
0000    258          .IIF NOT_BLANK  <MAX>, .WORD    MAX
0000    259          .IIF NOT_BLANK  <INVALID>, .WORD        INVALID
0000    260          'BASE'_PRM_BUFSIZ = 'BASE'_PRM_BUFSIZ + 6
0000    261          .ENDC
0000    262          .ENDM   PARAM
0000    263
0000    264          .MACRO  COUNT   TYPE,BITMAP=NO,WIDTH=8,BASE
0000    265          $$$TYP = TYPE & NMA$M_CNT_TYP
0000    266          .IIF IDN <BITMAP><YES>, $$$TYP = $$$TYP!<NMA$M_CNT_MAP>
0000    267          $$$WID = 0                      ; Set reserved mask width
0000    268          .IIF IDN <WIDTH><8>, $$$WID = <1@NMA$V_CNT_WID>
0000    269          .IIF IDN <WIDTH><16>, $$$WID = <2@NMA$V_CNT_WID>
0000    270          .IIF IDN <WIDTH><32>, $$$WID = <3@NMA$V_CNT_WID>
0000    271          .IIF EQ $$$WID, .ERROR          ; Invalid bit width value
0000    272          .WORD   NMA$M_CNT_COU!$$$WID!$$$TYP
0000    273          'BASE'_CNT_BUFSIZ = 'BASE'_CNT_BUFSIZ + 2 + <WIDTH/8>
0000    274          .IIF IDN <BITMAP><YES>, 'BASE'_CNT_BUFSIZ = 'BASE'_CNT_BUFSIZ + 2
0000    275          .ENDM   COUNT
0000    276
0000    277          .MACRO  SKIP    BIT,LOC,REG,CONTEXT=W,?L        ; SKIP FIELD
0000    278          BBC     #BIT,LOC,L              ; Br if field not present
0000    279          TST'CONTEXT     (REG)+          ; Skip next field
0000    280 L:
0000    281          .ENDM   SKIP
0000    282
0000    283 .MACRO  $DISPATCH,      INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
0000    284 SS:
0000    285
0000    286          .MACRO  $DSP1,$DSP1_1
0000    287                  .IRP    $DSP1_2,$DSP1_1
0000    288                          $DSP2   $DSP1_2
0000    289                  .ENDR
0000    290          .ENDM   $DSP1
0000    291
0000    292          .MACRO  $DSP2,$DSP2_1,$DSP2_2
0000    293                  .=<$DSP2_1-MN>*2 + S
0000    294                  .WORD   $DSP2_2-S
0000    295          .ENDM   $DSP2
0000    296
0000    297
0000    298          .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
0000    299                  $BND2   $BND1_1,$BND1_2
0000    300          .ENDM   $BND1
0000    301
0000    302          .MACRO  $BND2,$BND2_1,$BND2_2
0000    303                  .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
0000    304          .ENDM   $BND2
0000    305
0000    306          .MACRO  $BND    $BND_1,$BND_2
0000    307                  .IRP    $BND_3,<$BND_2>
0000    308                          $BND1   $BND_1,$BND_3
0000    309                  .ENDR
0000    310          .ENDM   $BND
0000    311
0000    312          .=0
0000    313 ZZ:
0000    314          $BND    GT,<VECTOR>
```

```
0000    315 MX:
0000    316         $BND    LT,<VECTOR>
0000    317 MN:
0000    318         .=SS
0000    319
0000    320 CASE'TYPE       INDX,#<MN-ZZ>,NMODE'<MX-MN>
0000    321 S:
0000    322         .REPT   MX-MN+1
0000    323         .WORD   <MX-MN>*2 + 2
0000    324         .ENDR
0000    325
0000    326         .=S
0000    327
0000    328         $DSP1   <<VECTOR>>
0000    329
0000    330         .=<MX-MN>*2 + S + 2
0000    331
0000    332 .ENDM   $DISPATCH
0000    333
```

```
                     0000   335 ;
                     0000   336 ; Overlays of IRP
                     0000   337 ;
                     0000   338         $DEFINI IRP
                     0000   339
00000021             0000   340 . = IRP$W_FUNC+1                        ; Overlay function word
                     0021   341
                     0021   342 $DEF     IRP$B_XDFUNC     .BLKB   1     ; DMP driver internal function code
                     0022   343
00000041             0022   344 . = IRP$Q_STATION+1                     ; Overlay Station address
                     0041   345
                     0041   346 $DEF     IRP$B_INDEX      .BLKB   1     ; Vector index for CDB
                     0042   347
00000042             0042   348 . = IRP$Q_STATION+2                     ; Overlay Station address
                     0042   349
                     0042   350 $DEF     IRP$W_QUOTA      .BLKW   1     ; Number of recieve buffers
                     0044   351
00000044             0044   352 . = IRP$Q_STATION+4                     ; Overlay ditto
                     0044   353
                     0044   354 $DEF     IRP$L_CDB        .BLKL   1     ; CDB address
                     0048   355
00000042             0048   356 . = IRP$Q_STATION+2                     ; Overlay ditto
                     0042   357
                     0042   358 $DEF     IRP$W_CSTATUS    .BLKW   1     ; Completion status
                     0044   359
0000003F             0044   360 . = IRP$L_MEDIA+7
                     003F   361
                     003F   362 $DEF     IRP$B_POLS       .BLKB   1     ; Polling substate
                     0040   363
                     0040   364 ;
                     0040   365 ; Define driver internal function codes stored in IRP$B_XDFUNC of IRP.
                     0040   366 ; NOTE: These are not really used as bit offsets - but as values.
                     0040   367 ;
                     0040   368         _VIELD  XD_FC,0,<-                ; Internal function codes
                     0040   369                 <INITT>,-                 ; Establish tributary
                     0040   370                 <INITC>,-                 ; Init the device
                     0040   371                 <STOPT>,-                 ; Shutdown tributary
                     0040   372                 <STOPC>,-                 ; Shutdown controller (device)
                     0040   373                 <CHGC>,-                  ; Change controller characteristics
                     0040   374                 <CHGT>,-                  ; Change trib characteristics
                     0040   375                 <RDMODEM>,-               ; Read line unit modem register
                     0040   376                 <RDTERR>,-                ; Read tributary error counters
                     0040   377                 <RDGERR>,-                ; Read global error counters
                     0040   378                 <RCTERR>,-                ; Read trib counters and clear
                     0040   379                 <RCGERR>,-                ; Read global counters and clear
                     0040   380                 <RDTSS>,-                 ; Read tributary status slot
                     0040   381                 <RDGSS>,-                 ; Read global status slot
                     0040   382                 <WTMODEM>,-               ; Write line unit modem register
                     0040   383                 <HALTT>,-                 ; DASSGN request to halt trib
                     0040   384                 <KILLT>,-                 ; DASSGN request to kill trib
                     0040   385                 <CANCEL>,-                ; CANCEL request to halt trib
                     0040   386                 <XMIT>,-                  ; Transmit request
                     0040   387                 <RECV>,-                  ; Receive request
                     0040   388                 <NUMSYNC>,-               ; Set number of sync chars
                     0040   389                 <POLSTAT>,-               ; Latch polling state
                     0040   390                 <POLSS>,-                 ; Fetch polling sub-state
                     0040   391                 >                         ;
```

```
                0040   392
                0040   393          $DEFEND IRP                     ; End of IRP overlays
```

```
                     0000   395 ;
                     0000   396 ; Definitions that follow the standard UCB fields
                     0000   397 ;
                     0000   398
                     0000   399          $DEFINI UCB     GLOBAL          ; Start of UCB definitions
00000090             0000   400 . = UCB$C_LENGTH                         ; Position at end of UCB
                     0090   401
                     0090   402 $DEF     UCB$L_XD_JNLBUF  .BLKL  1       ; Pointer to the journal buffer
                     0094   403 $DEF     UCB$Q_XD_QUEUES                 ; Message and I/O request queue heads
                     0094   404 $DEF     UCB$Q_XD_INPUTQ  .BLKQ  1       ; Request awaiting input port
                     009C   405 $DEF     UCB$Q_XD_XMT_REQ .BLKQ  1       ; Transmit I/O requests waiting mapping
                     00A4   406 $DEF     UCB$Q_XD_RCV_PND .BLKQ  1       ; Receive buffers given to device
                     00AC   407 $DEF     UCB$Q_XD_INFOUT  .BLKQ  1       ; Information Out queue
                     00B4   408 $DEF     UCB$Q_XD_POST    .BLKQ  1       ; Requests awaiting posting
                     00BC   409 $DEF     UCB$Q_XD_RCV_BUF .BLKQ  1       ; Free receive buffers (lookaside list)
00000006             00C4   410 UCB$C_XD_QUEUES = <.-UCB$Q_XD_QUEUES>/8 ; Number of queue heads
                     00C4   411
                     00C4   412 $DEF     UCB$L_XD_RCV_MAP .BLKL  MAX_RCV ; Receive mapping vector
                     00E0   413 $DEF     UCB$L_XD_XMT_MAP .BLKL  MAX_XMT ; Transmit mapping vector
                     00FC   414 $DEF     UCB$L_XD_RCV_PA  .BLKL  MAX_RCV_UV1 ; Receive mapping buffers physical
                     010C   415                                          ; address
                     010C   416 $DEF     UCB$L_XD_XMT_PA  .BLKL  MAX_XMT_UV1 ; Transmit mapping buffers physical
                     0114   417                                          ; address
                     0114   418 $DEF     UCB$L_XD_RCV_VA  .BLKL  MAX_RCV_UV1 ; Receive mapping buffers virtual
                     0124   419                                          ; address
                     0124   420 $DEF     UCB$L_XD_XMT_VA  .BLKL  MAX_XMT_UV1 ; Transmit mapping buffers virtual
                     012C   421                                          ; address
                     012C   422 $DEF     UCB$L_XD_UV1BUF  .BLKL  1       ; u-VAX I buffer area
                     0130   423 $DEF     UCB$B_XD_RCV_MAP .BLKB  1       ; Receive mapping in use flags
                     0131   424 $DEF     UCB$B_XD_XMT_MAP .BLKB  1       ; Transmit mapping in use flags
                     0132   425 $DEF     UCB$B_XD_RCV_MAX .BLKB  1       ; Maximum concurrent receives
                     0133   426 $DEF     UCB$B_XD_XMT_MAX .BLKB  1       ; Maximum concurrent transmits
                     0134   427 $DEF     UCB$B_XD_TRB_CNT .BLKB  1       ; Number of active tributaries
                     0135   428 $DEF     UCB$B_XD_XMT_TRB .BLKB  1       ; Maximum number of xmits per trib
                     0136   429 $DEF     UCB$W_XD_QUOTA   .BLKW  1       ; Receive buffer quota
                     0138   430
                     0138   431 $DEF     UCB$L_XD_VECS                   ; Start of vector areas
                     0138   432          ASSUME  MAX_DMP_TRIB GE MAX_DMV_TRIB ;allocate maximum slots
                     0138   433 $DEF     UCB$B_XD_TRIB_VEC .BLKB MAX_DMP_TRIB ; Tributary address vector
                     0158   434 $DEF     UCB$L_XD_PID_VEC .BLKL  MAX_DMP_TRIB ; PID Lookup vector
                     01D8   435 $DEF     UCB$W_XD_CHAN_VEC .BLKW MAX_DMP_TRIB ; Channel number lookup vector
                     0218   436 $DEF     UCB$L_XD_CDB_VEC .BLKL  MAX_DMP_TRIB ; CDB address vector
00000058             0298   437 UCB$C_XD_VECS = .-UCB$L_XD_VECS>/4       ; Number of longwords in vectors
                     0298   438
                     0298   439 $DEF     UCB$L_XD_RUN     .BLKL  1       ; Tribs that entered Run state
                     029C   440 $DEF     UCB$L_XD_HALT    .BLKL  1       ; Tribs that were forced to halt
                     02A0   441 $DEF     UCB$L_XD_PID     .BLKL  1       ; Starter's PID
                     02A4   442 $DEF     UCB$B_XD_MODE    .BLKB  1       ; Mode of operation in DMP11 format
                     02A5   443 $DEF     UCB$B_XD_INTIM   .BLKB  1       ; Input wait timer
                     02A6   444 $DEF     UCB$B_XD_OUTTIM  .BLKB  1       ; Output wait timer
                     02A7   445
                     02A7   446 $DEF     UCB$B_XD_SETPRM                 ; Start of parameter section
                     02A7   447 $DEF     UCB$B_XD_NMS     .BLKB  1       ; Number of sync chars (0=>default)
                     02A8   448 $DEF     UCB$B_XD_PRO     .BLKB  1       ; Protocol selection
                     02A9   449 $DEF     UCB$B_XD_DUP     .BLKB  1       ; Duplex mode
                     02AA   450 $DEF     UCB$B_XD_CON     .BLKB  1       ; Controller mode
                     02AB   451 $DEF     UCB$B_XD_BFN     .BLKB  1       ; Number of receive buffers
```

```
                02AC    452
                02AC    453 $DEF    UCB$W_XD_POLLPRM                    ; Start of polling parameter section
                02AC    454 $DEF    UCB$W_XD_SRT    .BLKW  1            ; Streaming tributary timer
                02AE    455 $DEF    UCB$W_XD_SLT    .BLKW  1            ; Delta timer
                02B0    456 $DEF    UCB$W_XD_DDT    .BLKW  1            ; Dead timer
                02B2    457 $DEF    UCB$W_XD_DLT    .BLKW  1            ; Poll delay time
                02B4    458 $DEF    UCB$W_XD_RTT    .BLKW  1            ; Retransmit timer
0000000F        02B6    459 UCB$C_XD_SETPRM = .-UCB$B_XD_SETPRM         ; Size of parameter section
                02B6    460
                02B6    461 $DEF    UCB$B_XD_FKB    .BLKB  FKB$C_LENGTH ; Fork process fork block
                02CE    462
                02CE    463 $DEF    UCB$C_XD_LENGTH                     ; Size of XDDRIVER UCB
                02CE    464
                02CE    465
000002C6        02CE    466 . = UCB$B_XD_FKB+FKB$L_FR3                  ; Position at R3 of fork block
                02C6    467
                02C6    468 $DEF    UCB$L_XD_SEL0   .BLKL  1            ; Saved SEL0 and SEL2 from interrupt
                02CA    469 $DEF    UCB$L_XD_SEL4   .BLKL  1            ; Saved SEL4 and SEL6 from interrupt
                02CE    470
                02CE    471 ;
                02CE    472 ; Define device status bits
                02CE    473 ;
                02CE    474         $VIELD  UCB,0,<-                    ; XDDRIVER UCB$W_DEVSTS bits
                02CE    475                 <XD_INITED,,M>,-            ; Device is initialized
                02CE    476                 <XD_INITING,,M>,-           ; Device is initing - waiting for Timeout
                02CE    477                 <XD_PTP,,M>,-               ; Device in point to point operation
                02CE    478                 <XD_FORK_PEND,,M>,-         ; Fork process scheduling in progress
                02CE    479                 <XD_DTRCLR,,M>,-            ; DTR modem signal is cleared
                02CE    480                 <XD_FORK_XMT,,M>,-          ; Transmit fork block in use
                02CE    481                 >                           ;
                02CE    482
                02CE    483         $DEFEND UCB                         ; End of UCB definitions
```

```
                     0000   485 ;
                     0000   486 ; Device register offsets and bit definitions
                     0000   487 ;
                     0000   488
                     0000   489         $DEFINI XD                          ; Start of status definitions
                     0000   490
                     0000   491
                     0000   492 $DEF     SEL0                                ; 1st word of CSR
                     0000   493 $DEF     BSEL0                               ; Byte select 0 / Control & status
00000001             0000   494                             .BLKB   1
                     0001   495
                     0001   496 ;
                     0001   497 ; Bit positions for device control/status register
                     0001   498 ;
                     0001   499
                     0001   500         _VIELD  XD_BSEL0,0,<-               ; Control & status bits
                     0001   501         <IEI,,M>,-                          ; Input interrupt enable
                     0001   502         <,3>,-                              ;
                     0001   503         <IEO,,M>,-                          ; Output interrupt enable
                     0001   504         <,2>,-                              ;
                     0001   505         <RQI,,M>,-                          ; Request input
                     0001   506         >
                     0001   507
                     0001   508 $DEF     BSEL1                               ; Byte select 1 / Control & status
00000002             0001   509                             .BLKB   1
                     0002   510
                     0002   511         _VIELD  XD_BSEL1,0,<-               ; Control & status bits
                     0002   512         <MAINT,,M>,-                        ; Maintenance request
                     0002   513         <,2>,-                              ;
                     0002   514         <LOOPB,,M>,-                        ; Loop back
                     0002   515         <,2>,-                              ;
                     0002   516         <MCLR,,M>,-                         ; Master clear
                     0002   517         <RUN,,M>,-                          ; RUN state
                     0002   518         >
                     0002   519
                     0002   520 $DEF     SEL2                                ; 2nd word of CSR
                     0002   521 $DEF     BSEL2                               ; Byte select 2 / Command register
00000003             0002   522                             .BLKB   1
                     0003   523
                     0003   524         _VIELD  XD_BSEL2,0,<-               ; Command register fields
                     0003   525         <TYPE,3,M>,-                        ; Command type code
                     0003   526         <Q22,,M>,-                          ; Q-Bus in 22 bit mode
                     0003   527         <RDI,,M>,-                          ; Ready input
                     0003   528         <ERR,,M>,-                          ; ** DRIVER DEFINED error indicator **
                     0003   529         <,1>,-                              ;
                     0003   530         <RDO,,M>,-                          ; Ready output
                     0003   531         >
                     0003   532
                     0003   533         _VIELD  XD_I,0,<-                   ; Define input command codes
                     0003   534         <RCV>,-                             ; Receive in      - 0
                     0003   535         <CONTROL>,-                         ; Control in      - 1
                     0003   536         <MODE_DEF>,-                        ; Mode definition - 2
                     0003   537         <,1>,-                              ; RESERVED
                     0003   538         <XMIT>,-                            ; Xmit in         - 4
                     0003   539         >
                     0003   540
                     0003   541 $DEF     BSEL3           .BLKB   1           ; Byte select 3 / Tributary address
```

```
0004    542
0004    543 $DEF    SEL4                            ; 3rd word of CSR
0004    544 $DEF    BSEL4           .BLKB   1       ; Byte select 4 / Misc.
0005    545
0005    546 $DEF    BSEL5           .BLKB   1       ; Byte select 5 / Misc.
0006    547
0006    548 $DEF    SEL6                            ; 4th word of CSR
0006    549 $DEF    BSEL6           .BLKB   1       ; TSS/GSS read/write requests
0007    550
0007    551         $VIELD XD_SEL6,0,<-             ; TSS/GSS control register
0007    552         <,5>,-                          ;
0007    553         <RTSS,,M>,-                     ; Read TSS
0007    554         <RCTS,,M>,-                     ; Read/clear TSS
0007    555         <WTSS,,M>,-                     ; Write TSS
0007    556         <ECOM,,M>,-                     ; Enable common pool
0007    557         <DCOM,,M>,-                     ; Disable common pool
0007    558         <,2>,-                          ;
0007    559         <UPOL,,M>,-                     ; Unlatch polling
0007    560         <LPOL,,M>,-                     ; Latch polling
0007    561         <,2>,-                          ;
0007    562         >
0007    563
0007    564         $VIELD  XD_CI,0,<-              ; SEL6 command codes
0007    565         <NOOP>,-                        ; NOOP request
0007    566         <ESTAB>,-                       ; Establish tributary
0007    567         <KILL>,-                        ; Kill tributary
0007    568         <ISTRT>,-                       ; Start tributary
0007    569         <MAINT>,-                       ; Start trib in MOP mode
0007    570         <HALT>,-                        ; Halt tributary
0007    571         <,10>,-                         ; RESERVED
0007    572         <RDMODEM>,-                     ; Read Modem request
0007    573         <WTMODEM>,-                     ; Write Modem request
0007    574         <INTDIAG>-                      ; Interface diagnostics
0007    575         >
0007    576
0007    577 $DEF    BSEL7                           ; Byte select 7 / Misc.
00000008 0007   578                     .BLKB   1
0008    579
0008    580 $DEF    SEL10           .BLKW   1       ; Byte select 10 - character count
000A    581                                         ;   for DMV only
000A    582
000A    583         $DEFEND XD                      ; End of device register definitions
```

```
                     0000   585 ;
                     0000   586 ; XDDRIVER CDB definitions
                     0000   587 ;
                     0000   588          $DEFINI CDB     GLOBAL
00000000             0000   589 .=0
                     0000   590 $DEF     CDB_L_DEVDEPEND .BLKL   1       ; Circuit dependent status
                     0004   591 $DEF     CDB_L_AST       .BLKL   1       ; Attention AST list
                     0008   592 $DEF     CDB_W_SIZE      .BLKW   1       ; Size of CDB
                     000A   593 $DEF     CDB_B_TYPE      .BLKB   1       ; Type of structure
                     000B   594 $DEF     CDB_B_POL       .BLKB   1       ; Polling state
                     000C   595 $DEF     CDB_W_STS       .BLKW   1       ; Circuit status
                     000E   596 $DEF     CDB_B_TRB_ADDR  .BLKB   1       ; Tributary address
                     000F   597 $DEF     CDB_B_XMT_CNT   .BLKB   1       ; Number of xmits outstanding
                     0010   598
                     0010   599 $DEF     CDB_Q_QUEUES                    ; Message and I/O request queue heads
                     0010   600 $DEF     CDB_Q_XMT_REQ   .BLKQ   1       ; Transmit requests awaiting RUN
                     0018   601 $DEF     CDB_Q_RCV_REQ   .BLKQ   1       ; Receive I/O requests awaiting message
                     0020   602 $DEF     CDB_Q_RCV_MSG   .BLKQ   1       ; Receive buffers containing messages
                     0028   603 $DEF     CDB_Q_XMT_PND   .BLKQ   1       ; Transmit I/Os given to device
                     0030   604 $DEF     CDB_Q_INFOUT    .BLKQ   1       ; Information Out queue
00000005             0038   605 CDB_C_QUEUES = <.-CDB_Q_QUEUES>/8          ; Number of queue heads in CDB
                     0038   606
                     0038   607 $DEF     CDB_L_CNTS                      ; Start of counters area of CDB
                     0038   608 $DEF     CDB_L_BRC       .BLKL   1       ; Receive byte count
                     003C   609 $DEF     CDB_L_BSN       .BLKL   1       ; Transmit byte count
                     0040   610 $DEF     CDB_L_DMR       .BLKL   1       ; Receive message count
                     0044   611 $DEF     CDB_L_DMS       .BLKL   1       ; Transmit message count
00000010             0048   612 CDB_C_CNTS = .-CDB_L_CNTS                  ; Size of counter section of CDB
                     0048   613
                     0048   614 $DEF     CDB_L_PID       .BLKL   1       ; Starter's process ID
                     004C   615 $DEF     CDB_W_CHAN      .BLKW   1       ; Channel number associated with this CDB
                     004E   616
                     004E   617 $DEF     CDB_B_SETPRM                    ; Start of settable parameters
                     004E   618 $DEF     CDB_B_MRB       .BLKB   1       ; Receive buffer count
                     004F   619 $DEF     CDB_B_MST       .BLKB   1       ; Maintenance State given by P2 buffer
                     0050   620
                     0050   621 $DEF     CDB_W_POLLPRM                   ; Start of Poll parameter section of CDB
                     0050   622 $DEF     CDB_W_DEL_TIMER .BLKW   1       ; Delay timer
                     0052   623 $DEF     CDB_B_POL_QACT  .BLKB   1       ; Active trib Q value
                     0053   624 $DEF     CDB_B_POL_RACT  .BLKB   1       ; Active trib R value
                     0054   625 $DEF     CDB_B_POL_QIN   .BLKB   1       ; Inactive trib Q value
                     0055   626 $DEF     CDB_B_POL_RIN   .BLKB   1       ; Inactive triv R value
                     0056   627 $DEF     CDB_B_POL_QDYI  .BLKB   1       ; Dying Q value
                     0057   628 $DEF     CDB_B_POL_RDYI  .BLKB   1       ; Dying R value
                     0058   629 $DEF     CDB_B_TO_INACT  .BLKB   1       ; Number of NDM to Inactive
                     0059   630 $DEF     CDB_B_TO_DYING  .BLKB   1       ; Number of no responses to Dying
                     005A   631 $DEF     CDB_B_TO_DEAD   .BLKB   1       ; Number of no responses to Dead
                     005B   632 $DEF     CDB_B_MAX_MSGS  .BLKB   1       ; Maximum number of xmits per selection
                     005C   633 $DEF     CDB_W_SEL_TIMER .BLKW   1       ; Selection timer
                     005E   634 $DEF     CDB_W_BAB_TIMER .BLKW   1       ; Babbling trib timer
00000012             0060   635 CDB_C_SETPRM = .-CDB_B_SETPRM             ; Size of settable param section of CDB
                     0060   636
                     0060   637 $DEF     CDB_C_LENGTH                    ; Size of XDDRIVER CDB
                     0060   638
                     0060   639 ;
                     0060   640 ; Define status bits used in CDB_B_STS
                     0060   641 ;
```

```
                                    K 3
0060    642
0060    643         _VIELD  CD_TS,0,<-                  ; Tributary status bits for CDB_W_STS
0060    644                 <ESTAB,,M>,-                ; Tributary established
0060    645                 <MOP,,M>,-                  ; Tributary is in MOP mode
0060    646                 <CANCEL,,M>,-               ; Cancel I/O in progress
0060    647                 <BUFRET,,M>,-               ; Buffers have been returned
0060    648                 <BUFQUO,,M>,-               ; Buffer quota management needed
0060    649                 <START,,M>,-                ; Tributary is started
0060    650                 >                           ;
0060    651
0060    652         $DEFEND CDB
```

```
                     0000   654 ;
                     0000   655 ; Receive buffer definition
                     0000   656 ;
                     0000   657          $DEFINI RCV
00000000             0000   658 .=0
                     0000   659 $DEF     RCV_L_LINK      .BLKL   2       ; Forward and backward queue links
                     0008   660 $DEF     RCV_W_BLKSIZE   .BLKW   1       ; Total block size
                     000A   661 $DEF     RCV_B_BLKTYPE   .BLKB   1       ; Block type
                     000B   662 $DEF     RCV_B_MAPSLOT   .BLKB   1       ; Mapping slot number
                     000C   663 $DEF     RCV_L_BACC      .BLKL   1       ; Buffer address / character count
                     0010   664 $DEF     RCV_B_BAHI      .BLKB   1       ; Buffer address hi 6 bits (Q22 mode)
                     0048   666 $DEF     RCV_T_DATA                      ; Receive data
                     0048   667
                     0048   668          $DEFEND RCV
                     0000   669
                     0000   670 ;
                     0000   671 ; P2 buffer header definition
                     0000   672 ;
                     0000   673          $DEFINI P2B
00000000             0000   674 .=0
                     0000   675 $DEF     P2B_L_POINTER   .BLKL   1       ; Pointer to start of data
                     0004   676 $DEF     P2B_L_BUFFER    .BLKL   1       ; Address of user's data buffer
                     0008   677 $DEF     P2B_W_SIZE      .BLKW   1       ; Size of P2 buffer
                     000A   678 $DEF     P2B_B_TYPE      .BLKB   1       ; Type of structure
                     000B   679 $DEF     P2B_B_SPARE     .BLKB   1       ; Spare byte
                     000C   680 $DEF     P2B_C_LENGTH                    ; Size of P2 buffer header
                     000C   681 $DEF     P2B_T_DATA                      ; Start of data
                     000C   682
                     000C   683          $DEFEND P2B
```

```
                        .SBTTL  Standard tables
                ;
                ; Driver prologue table
                ;

                DPTAB   -                                       ; DPT-creation macro
                        END=XD_END,-                            ; End of driver label
                        ADAPTER=UBA,-                           ; Adapter type
                        UCBSIZE=<UCB$C_XD_LENGTH>,-             ; Length of UCB
                        NAME=XDDRIVER,-                         ; Driver name

                DPT_STORE INIT                                  ; Start of load
                                                                ; initialization table
                DPT_STORE UCB,UCB$B_FIPL,B,8                    ; Device fork IPL
                DPT_STORE UCB,UCB$B_DIPL,B,21                   ; Device interrupt IPL
                DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics:
                        DEV$M_NET!-                             ;   network device
                        DEV$M_SHR!-                             ;   shared device
                        DEV$M_AVL!-                             ;   available device
                        DEV$M_IDV!-                             ;   input device
                        DEV$M_ODV>                              ;   output device
                DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM         ; Device class - synch comm
                DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_DMP11         ; Device type

                DPT_STORE REINIT                                ; Start of reload
                                                                ; initialization table
                DPT_STORE DDB,DDB$L_DDT,D,XD$DDT                ; Address of DDT
                DPT_STORE CRB,-                                 ; Address of device
                        CRB$L_INTD+VEC$L_UNITINIT,-             ; unit initialization
                        D,UNIT_INIT                             ; routine
                DPT_STORE CRB,-                                 ; Address of CSR RDI
                        CRB$L_INTD+4,D,RDI_INTRPT               ; interrupt routine
                DPT_STORE CRB,-                                 ; Address of CSR RDO
                        CRB$L_INTD2+4,D,RDO_INTRPT              ; interrupt routine
                DPT_STORE UCB,UCB$L_FPC,D,FORK_PROC             ; Set up timeout vector

                DPT_STORE END                                   ; End of initialization
                                                                ; tables

                ;
                ; Driver dispatch table
                ;

                DDTAB   -                                       ; DDT-creation macro
                        DEVNAM=XD,-                             ; Name of device
                        START=STARTIO,-                         ; Start I/O routine
                        FUNCTB=XD_FUNCTABLE,-                   ; FDT address
                        CANCEL=CANCEL,-                         ; Cancel I/O routine
                        REGDMP=REG_DUMP,-                       ; Register dump routine
                        DIAGBF=<36*12>,-                        ; Diagnostic buffer sizing
                        ALTSTART=ALT_START                      ; Alternate start I/O

                ;
                ; Function dispatch table
                ;
```

```
                        ;
                        ; Standard tables
                        ;
                        ;++
                        ; XD_FUNCTABLE - Function table for XD driver
                        ;
                        ; This table defines the functions which are valid for the XD driver.
                        ;--
                        ;
0038   742 XD_FUNCTABLE:
0038   743         FUNCTAB ,-                              ; FDT for driver
0038   744                 <READVBLK,-                     ; Valid I/O functions:
0038   745                 READLBLK,-                      ; Read virtual
0038   746                 READPBLK,-                      ; Read logical
0038   747                 WRITEVBLK,-                     ; Read physical
0038   748                 WRITELBLK,-                     ; Write virtual
0038   749                 WRITEPBLK,-                     ; Write logical
0038   750                 SETMODE,-                       ; Write physical
0038   751                 SENSEMODE,-                     ; Set device mode
0038   752                 SETCHAR>                        ; Sense mode
0040   753                                                 ; Set device chars.
0040   754         FUNCTAB ,-                              ; Buffered functions:
0040   755                 <READVBLK,-                     ; Read virtual
0040   756                 READLBLK,-                      ; Read logical
0040   757                 READPBLK,-                      ; Read physical
0040   758                 WRITEVBLK,-                     ; Write virtual
0040   759                 WRITELBLK,-                     ; Write logical
0040   760                 WRITEPBLK,-                     ; Write physical
0040   761                 SETMODE,-                       ; Set device mode
0040   762                 SENSEMODE,-                     ; Sense mode
0040   763                 SETCHAR>                        ; Set device chars.
0048   764
0048   765         FUNCTAB RCV_FDT,-                       ; FDT read routine for
0048   766                 <READVBLK,-                     ; read virtual,
0048   767                 READLBLK,-                      ; read logical,
0048   768                 READPBLK>                       ; and read physical.
0054   769
0054   770         FUNCTAB XMT_FDT,-                       ; FDT write routine for
0054   771                 <WRITEVBLK,-                    ; write virtual,
0054   772                 WRITELBLK,-                     ; write logical,
0054   773                 WRITEPBLK>                      ; and write physical.
0060   774
0060   775         FUNCTAB SETMODE_FDT,-                   ; FDT set mode routine
0060   776                 <SETMODE,SETCHAR>               ; for set mode, set char.
006C   777
006C   778         FUNCTAB SENSEMODE_FDT,-                 ; FDT sense mode routine
006C   779                 <SENSEMODE>                     ; for sensemode
```

```
                     0078   781          .SBTTL  P2 buffer verification tables
                     0078   782
                     0078   783 ;
                     0078   784 ; Define P2 buffer verification offsets
                     0078   785 ;
                     0078   786          $DEFINI PARAM
                     0000   787
                     0000   788          _VIELD  PRM,0,<-                        ; Parameter bits and sizes
                     0000   789                  <TYPE,12,M>,-                   ; Parameter type
                     0000   790                  <MIN,1,M>,-                     ; Parameter minimum value
                     0000   791                  <MAX,1,M>,-                     ; Parameter maximum value
                     0000   792                  <INVALID,1,M>,-                 ; Parameter invalid flags
                     0000   793                  >
                     0000   794
                     0000   795          _VIELD  OFF,0,<-                        ; Offset word fields
                     0000   796                  <VALUE,14,M>,-                  ; Offset value
                     0000   797                  <WIDTH,2,M>,-                   ; Size of field in structure
                     0000   798                  >
                     0000   799
                     0000   800          $DEFEND PARAM
                     0078   801
                     0078   802 ;
                     0078   803 ; Define UCB parameters
                     0078   804 ;
                00000000  0078   805 LINE_PRM_BUFSIZ = 0                         ; Line parameter buffer size
                     0078   806 LINE_PARAM:                                     ; Start of line parameters
                     0078   807
                     0078   808          PARAM   NMA$C_PCLI_PRO,-                ; Protocol selection
                     0078   809                  OFFSET=UCB$B_XD_PRO,WIDTH=B,-
                     0078   810                  MAX=NMA$C_LINPR_DMC,-
                     0078   811                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
                     0078   812                  BASE=LINE
                     0080   813
                     0080   814          PARAM   NMA$C_PCLI_DUP,-                ; Duplex mode
                     0080   815                  OFFSET=UCB$B_XD_DUP,WIDTH=B,-
                     0080   816                  MAX=NMA$C_DPX_HAL,-
                     0080   817                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
                     0080   818                  BASE=LINE
                     0088   819
                     0088   820          PARAM   NMA$C_PCLI_CON,-                ; Conroller mode
                     0088   821                  OFFSET=UCB$B_XD_CON,WIDTH=B,-
                     0088   822                  MAX=NMA$C_LINCN_LOO,-
                     0088   823                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
                     0088   824                  BASE=LINE
                     0090   825
                     0090   826          PARAM   NMA$C_PCLI_BFN,-                ; Number of receives
                     0090   827                  OFFSET=UCB$B_XD_BFN,WIDTH=B,-
                     0090   828                  MIN=1,MAX=255,-
                     0090   829                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
                     0090   830                  BASE=LINE
                     009A   831
                     009A   832          PARAM   NMA$C_PCLI_BUS,-                ; Buffer size
                     009A   833                  OFFSET=UCB$W_DEVBUFSIZ,WIDTH=W,-
                     009A   834                  MIN=1,MAX=MAX_BUFSIZ,-
                     009A   835                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
                     009A   836                  BASE=LINE
                     00A4   837
```

```
                     838          PARAM   NMA$C_PCLI_SLT,-                        ; Delta timer
                     839                  OFFSET=UCB$W_XD_SLT,WIDTH=W,-
                     840                  MIN=50,-
                     841                  BASE=LINE
                     842
                     843          PARAM   NMA$C_PCLI_DDT,-                        ; Dead timer
                     844                  OFFSET=UCB$W_XD_DDT,WIDTH=W,-
                     845                  MIN=50,-
                     846                  BASE=LINE
                     847
                     848          PARAM   NMA$C_PCLI_DLT,-                        ; Delay timer
                     849                  OFFSET=UCB$W_XD_DLT,WIDTH=W,-
                     850                  BASE=LINE
                     851
                     852          PARAM   NMA$C_PCLI_SRT,-                        ; Streaming trib timer
                     853                  OFFSET=UCB$W_XD_SRT,WIDTH=W,-
                     854                  MIN=50,-
                     855                  BASE=LINE
                     856
                     857          PARAM   NMA$C_PCLI_NMS,-                        ; Number of sync chars
                     858                  OFFSET=UCB$B_XD_NMS,WIDTH=B,-
                     859                  MIN=8,MAX=255,-
                     860                  BASE=LINE
                     861
                     862          PARAM   NMA$C_PCLI_RTT,-                        ; Retransmit timer
                     863                  OFFSET=UCB$W_XD_RTT,WIDTH=W,-
                     864                  MIN=50,-
                     865                  BASE=LINE
                     866
                     867          PARAM                                           ; End of table
                     868 ;
                     869 ; Define CDB parameters
                     870 ;
00000000             871 TRIB_PRM_BUFSIZ  = 0                                     ; Trib parameter buffer size
                     872 TRIB_PARAM:                                             ; Start of trib parameters
                     873
                     874          PARAM   NMA$C_PCCI_TRI,-                        ; Trib address
                     875                  OFFSET=CDB_B_TRB_ADDR,WIDTH=B,-
                     876                  MIN=1,-
                     877                  INVALID=XM$M_STS_ACTIVE,-               ; Trib can't be established
                     878                  BASE=TRIB
                     879
                     880          PARAM   NMA$C_PCCI_MRB,-                        ; Maximum receive buffers
                     881                  OFFSET=CDB_B_MRB,WIDTH=B,-
                     882                  MIN=1,-
                     883                  INVALID=XM$M_STS_ACTIVE,-               ; Trib can't be STARTING
                     884                  BASE=TRIB
                     885
                     886          ASSUME  NMA$C_STATE_ON EQ 0
                     887          PARAM   NMA$C_PCCI_MST,-                        ; Maintenance state
                     888                  OFFSET=CDB_B_MST,WIDTH=B,-
                     889                  MAX=NMA$C_STATE_OFF,-
                     890                  INVALID=XM$M_STS_ACTIVE,-               ; Trib can't be established
                     891                  BASE=TRIB
                     892
                     893          PARAM   NMA$C_PCCI_POL,-                        ; Latch polling state
                     894                  OFFSET=CDB_B_POL,WIDTH=B,-
```

```
00E2    895                 MAX=NMA$C_CIRPST_DED,-
00E2    896                 BASE=TRIB
00E8    897
00E8    898         PARAM   NMA$C_PCCI_TRT,-                ; Transmit delay timer
00E8    899                 OFFSET=CDB_W_DEL_TIMER,WIDTH=W,-
00E8    900                 BASE=TRIB
00EC    901
00EC    902         PARAM   NMA$C_PCCI_ACB,-                ; Active base
00EC    903                 OFFSET=CDB_B_POL_QACT,WIDTH=B,-
00EC    904                 BASE=TRIB
00F0    905
00F0    906         PARAM   NMA$C_PCCI_ACI,-                ; Active increment
00F0    907                 OFFSET=CDB_B_POL_RACT,WIDTH=B,-
00F0    908                 BASE=TRIB
00F4    909
00F4    910         PARAM   NMA$C_PCCI_IAB,-                ; Inactive base
00F4    911                 OFFSET=CDB_B_POL_QIN,WIDTH=B,-
00F4    912                 BASE=TRIB
00F8    913
00F8    914         PARAM   NMA$C_PCCI_IAI,-                ; Inactive increment
00F8    915                 OFFSET=CDB_B_POL_RIN,WIDTH=B,-
00F8    916                 BASE=TRIB
00FC    917
00FC    918         PARAM   NMA$C_PCCI_DYB,-                ; Dying base
00FC    919                 OFFSET=CDB_B_POL_QDYI,WIDTH=B,-
00FC    920                 BASE=TRIB
0100    921
0100    922         PARAM   NMA$C_PCCI_DYI,-                ; Dying increment
0100    923                 OFFSET=CDB_B_POL_RDYI,WIDTH=B,-
0100    924                 BASE=TRIB
0104    925
0104    926         PARAM   NMA$C_PCCI_IAT,-                ; Inactive threshold
0104    927                 OFFSET=CDB_B_TO_INACT,WIDTH=B,-
0104    928                 BASE=TRIB
0108    929
0108    930         PARAM   NMA$C_PCCI_DYT,-                ; Dying threshold
0108    931                 OFFSET=CDB_B_TO_DYING,WIDTH=B,-
0108    932                 BASE=TRIB
010C    933
010C    934         PARAM   NMA$C_PCCI_DTH,-                ; Dead threshold
010C    935                 OFFSET=CDB_B_TO_DEAD,WIDTH=B,-
010C    936                 BASE=TRIB
0110    937
0110    938         PARAM   NMA$C_PCCI_MTR,-                ; Max abutted xmits
0110    939                 OFFSET=CDB_B_MAX_MSGS,WIDTH=B,-
0110    940                 MIN=1,-
0110    941                 MAX=100,-
0110    942                 BASE=TRIB
0118    943
0118    944         PARAM   NMA$C_PCCI_BBT,-                ; Babbling trib timer
0118    945                 OFFSET=CDB_W_BAB_TIMER,WIDTH=W,-
0118    946                 MIN=50,-
0118    947                 BASE=TRIB
011E    948
011E    949         PARAM   NMA$C_PCCI_RTT,-                ; Selection timer
011E    950                 OFFSET=CDB_W_SEL_TIMER,WIDTH=W,-
011E    951                 MIN=50,-
```

```
                     011E   952                BASE=TRIB
                     0124   953
                     0124   954        PARAM                                   ; End of trib tables
                     0126   955 ;
                     0126   956 ; DEFAULT TRIBUTARY PARAMETERS
                     0126   957 ;
                     0126   958 DEF_TRIB_PARAM::
                     0126   959        ASSUME  CDB_B_MRB EQ CDB_B_SETPRM
                     0126   960        ASSUME  CDB_B_MST EQ CDB_B_MRB+1
                     0126   961        ASSUME  CDB_W_DEL_TIMER EQ CDB_B_MST+1
                     0126   962        ASSUME  CDB_B_POL_QACT EQ CDB_W_DEL_TIMER+2
                     0126   963        ASSUME  CDB_B_POL_RACT EQ CDB_B_POL_QACT+1
                     0126   964        ASSUME  CDB_B_POL_QIN EQ CDB_B_POL_RACT+1
                     0126   965        ASSUME  CDB_B_POL_RIN EQ CDB_B_POL_QIN+1
                     0126   966        ASSUME  CDB_B_POL_QDYI EQ CDB_B_POL_RIN+1
                     0126   967        ASSUME  CDB_B_POL_RDYI EQ CDB_B_POL_QDYI+1
                     0126   968        ASSUME  CDB_B_TO_INACT EQ CDB_B_POL_RDYI+1
                     0126   969        ASSUME  CDB_B_TO_DYING EQ CDB_B_TO_INACT+1
                     0126   970        ASSUME  CDB_B_TO_DEAD EQ CDB_B_TO_DYING+1
                     0126   971        ASSUME  CDB_B_MAX_MSGS EQ CDB_B_TO_DEAD+1
                     0126   972        ASSUME  CDB_W_SEL_TIMER EQ CDB_B_MAX_MSGS+1
                     0126   973        ASSUME  CDB_W_BAB_TIMER EQ CDB_W_SEL_TIMER+2
               FF    0126   974        .BYTE   INF_RCV                         ; Default # of receive buffers
               01    0127   975        .BYTE   NMA$C_STATE_OFF                 ; Default maint state is OFF
             0000    0128   976        .WORD   0                               ; Transmit delay timer
               FF    012A   977        .BYTE   255                             ; Q for active
               00    012B   978        .BYTE   0                               ; R for active
               00    012C   979        .BYTE   0                               ; Q for inactive
               40    012D   980        .BYTE   64                              ; R for inactive
               00    012E   981        .BYTE   0                               ; Q for dying
               10    012F   982        .BYTE   16                              ; R for dying
               08    0130   983        .BYTE   8                               ; Inactive threshold
               02    0131   984        .BYTE   2                               ; Dying threshold
               03    0132   985        .BYTE   3                               ; Dead threshold
               04    0133   986        .BYTE   4                               ; Maximum abutted messages
             0BB8    0134   987        .WORD   3000                            ; Selection timer (dummy entry)
             1770    0136   988        .WORD   6000                            ; Babble timer in Msec
         00000012    0138   989        DEF_TRIB_PARAMSZ = .-DEF_TRIB_PARAM
                     0138   990        ASSUME  CDB_C_SETPRM EQ DEF_TRIB_PARAMSZ
                     0138   991 ;
                     0138   992 ; Default line parameter values
                     0138   993 ;
                     0138   994 DEF_LINE_PARAM::
                     0138   995        ASSUME  UCB$B_XD_NMS EQ UCB$B_XD_SETPRM
                     0138   996        ASSUME  UCB$B_XD_PRO EQ UCB$B_XD_NMS+1
                     0138   997        ASSUME  UCB$B_XD_DUP EQ UCB$B_XD_PRO+1
                     0138   998        ASSUME  UCB$B_XD_CON EQ UCB$B_XD_DUP+1
                     0138   999        ASSUME  UCB$B_XD_BFN EQ UCB$B_XD_CON+1
                     0138  1000        ASSUME  UCB$W_XD_SRT EQ UCB$B_XD_BFN+1
                     0138  1001        ASSUME  UCB$W_XD_SLT EQ UCB$W_XD_SRT+2
                     0138  1002        ASSUME  UCB$W_XD_DDT EQ UCB$W_XD_SLT+2
                     0138  1003        ASSUME  UCB$W_XD_DLT EQ UCB$W_XD_DDT+2
                     0138  1004        ASSUME  UCB$W_XD_RTT EQ UCB$W_XD_DLT+2
               00    0138  1005        .BYTE   0                               ; Number of sync chars (default)
               00    0139  1006        .BYTE   NMA$C_LINPR_POI                 ; Protocol is point-point
               00    013A  1007        .BYTE   NMA$C_DPX_FUL                   ; Duplex is full
               00    013B  1008        .BYTE   NMA$C_LINCN_NOR                 ; Controller mode is normal
```

```
                     013C  1009          .BYTE   0                               ; Number of receive buffers
                     013D  1010          .WORD   6000                            ; Streaming trib timer in Msec
                     013F  1011          .WORD   50                              ; Delta timer in Msec
                     0141  1012          .WORD   10000                           ; Dead timer
                     0143  1013          .WORD   0                               ; Poll delay timer in Msec
                     0145  1014          .WORD   3000                            ; Selection timer in Msec
                     0147  1015          DEF_LINE_PARAMSZ = .-DEF_LINE_PARAM
                     0147  1016          ASSUME  OCB$C_XD_SETPRM EQ DEF_LINE_PARAMSZ
                     0147  1017 ;
                     0147  1018 ; Polling sub-state return table
                     0147  1019 ;
                     0147  1020          ASSUME  NMA$C_CIRPST_ACT NE 0
                     0147  1021          ASSUME  NMA$C_CIRPST_INA NE 0
                     0147  1022          ASSUME  NMA$C_CIRPST_DIE NE 0
                     0147  1023          ASSUME  NMA$C_CIRPST_DED NE 0
                     0147  1024 POLSS_TBL:
                  02 0147  1025          .BYTE   NMA$C_CIRPST_ACT                ; Active
                  03 0148  1026          .BYTE   NMA$C_CIRPST_INA                ; Inactive
                  00 0149  1027          .BYTE   0                               ; Unknown
                  00 014A  1028          .BYTE   0                               ; Unknown
                  04 014B  1029          .BYTE   NMA$C_CIRPST_DIE                ; Dying
                  00 014C  1030          .BYTE   0                               ; Unknown
                  00 014D  1031          .BYTE   0                               ; Unknown
                  05 014E  1032          .BYTE   NMA$C_CIRPST_DED                ; Dead
                     014F  1033
                     014F  1034 ;
                     014F  1035 ;  Maximum buffer size allowed - assumed to be MAX_BUFSIZ (16KB), but
                     014F  1036 ;  gets overlaid if the processor is a u-VAX I to be MAX_BUFSIZ_UV1.
                     014F  1037 ;
                     014F  1038 MAX_W_BUFFER:
                3FFF 014F  1039          .WORD   MAX_BUFSIZ                      ; Assume 16kb buffers
                     0151  1040 MAX_W_TRIB:
                0020 0151  1041          .WORD   MAX_DMP_TRIB                    ; Assume maximum tribs for DMP11
                     0153  1042 ;
                     0153  1043 ;
                     0153  1044 ; Line counter type codes
                     0153  1045 ;
                00000000 0153  1046 LINE_CNT_BUFSIZ = 0                          ; Size of ctrl counter buffer
                     0153  1047 LINE_COUNTER:
                     0153  1048          COUNT   NMA$C_CTLIN_RPE,-               ; Remote process errors
                     0153  1049                  BITMAP=YES,WIDTH=8,BASE=LINE
                     0155  1050
                     0155  1051          COUNT   NMA$C_CTLIN_LPE,-               ; Local process errors
                     0155  1052                  BITMAP=YES,WIDTH=8,BASE=LINE
                     0157  1053
                     0157  1054 ;
                     0157  1055 ; Tributary counter type codes
                     0157  1056 ;
                     0157  1057 ; NOTE: The first set of trib counters are kept by the driver, since
                     0157  1058 ; the device only keeps 16 bit counters for these counters.
                     0157  1059 ;
                00000000 0157  1060 TRIB_CNT_BUFSIZ = 0                          ; Size of return buffer for trib cou
                     0157  1061 TRIB_COUNTER:
                     0157  1062          COUNT   NMA$C_CTCIR_BRC,-               ; Bytes received
                     0157  1063                  WIDTH=32,BASE=TRIB
                     0159  1064
                     0159  1065          COUNT   NMA$C_CTCIR_BSN,-               ; Bytes sent
```

```
                                        WIDTH=32,BASE=TRIB

                        COUNT   NMA$C_CTCIR_DBR,-              ; Data blocks received
                                WIDTH=32,BASE=TRIB

                        COUNT   NMA$C_CTCIR_DBS,-              ; Data blocks sent
                                WIDTH=32,BASE=TRIB

TRIB_COUNTER1:
                        COUNT   NMA$C_CTCIR_SIE,-              ; Selection intervals elapsed
                                WIDTH=16,BASE=TRIB

                        COUNT   NMA$C_CTCIR_DEO,-              ; Data errors outbound
                                BITMAP=YES,WIDTH=8,BASE=TRIB

                        COUNT   NMA$C_CTCIR_DEI,-              ; Data errors inbound
                                BITMAP=YES,WIDTH=8,BASE=TRIB

                        COUNT   NMA$C_CTCIR_LBE,-              ; Local buffer errors
                                BITMAP=YES,WIDTH=8,BASE=TRIB

                        COUNT   NMA$C_CTCIR_RBE,-              ; Remote buffer errors
                                BITMAP=YES,WIDTH=8,BASE=TRIB

                        COUNT   NMA$C_CTCIR_SLT,-              ; Selection timeouts
                                BITMAP=YES,WIDTH=8,BASE=TRIB

                        COUNT   NMA$C_CTCIR_LRT,-              ; Local reply timeouts
                                WIDTH=8,BASE=TRIB

                        COUNT   NMA$C_CTCIR_RRT,-              ; Remote reply timeouts
                                WIDTH=8,BASE=TRIB
```

```
                                                H 4
                                                016F  1100          .SBTTL  UNIT_INIT,  Unit initialization routine
                                                016F  1101
                                                016F  1102 ;++
                                                016F  1103 ; UNIT_INIT - Readies unit for I/O operations
                                                016F  1104 ;
                                                016F  1105 ; Functional description:
                                                016F  1106 ;
                                                016F  1107 ;       The operating system calls this routine after calling the
                                                016F  1108 ;       controller initialization routine:
                                                016F  1109 ;
                                                016F  1110 ;               at system startup
                                                016F  1111 ;               during driver loading
                                                016F  1112 ;               during recovery from a power failure
                                                016F  1113 ;
                                                016F  1114 ;       The unit is put online.  If power has failed, the unit will
                                                016F  1115 ;       be forced to timeout.
                                                016F  1116 ;
                                                016F  1117 ; Inputs:
                                                016F  1118 ;
                                                016F  1119 ;       R4 = CSR address
                                                016F  1120 ;       R5 = UCB address
                                                016F  1121 ;
                                                016F  1122 ; Outputs:
                                                016F  1123 ;
                                                016F  1124 ;       R0 destroyed
                                                016F  1125 ;
                                                016F  1126 ;--
                                                016F  1127
                                                016F  1128 UNIT_INIT:                                   ; Initialize unit
                                          1F    BB  016F  1129          PUSHR   #^M<R0,R1,R2,R3,R4>     ; Save all registers
                                    64    A5    10  A8  0171  1130          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; Set unit online
                                      0B  A5    90  0175  1131          MOVB    UCB$B_FIPL(R5),-                ; Set the fork block FIPL
                                    02C1  C5        0178  1132                  UCB$B_XD_FKB+FKB$B_FIPL(R5)     ;
                                          08    90  017B  1133          MOVB    #DYN$C_FRK,-                    ; Set structure to FORK BLOCK
                                    02C0  C5        017D  1134                  UCB$B_XD_FKB+FKB$B_TYPE(R5)     ;
                                       226E   30  0180  1135          BSBW    INIT_UCB                        ; Initialize the UCB
                                              0183  1136          ;
                                              0183  1137          ; For u-VAX I, we will have to allocate a physically contiguous buffer
                                              0183  1138          ; area for performing I/O on the DMV.
                                              0183  1139          ;
                                              0183  1140          CPUDISP <<790,INIT_OTHERS>,-
                                              0183  1141                   <780,INIT_OTHERS>,-
                                              0183  1142                   <750,INIT_OTHERS>,-
                                              0183  1143                   <730,INIT_OTHERS>,-
                                              0183  1144                   <UV1,INIT_UV1>>
                                              019D  1145
                                              019D  1146 INIT_UV1:
                                0400  8F    B0  019D  1147          MOVW    #MAX_BUFSIZ_UV1,MAX_W_BUFFER    ; Set maximum size buffer
                                  AA  AF    10  B0  01A3  1148          MOVW    #MAX_DMV_TRIB,MAX_W_TRIB        ; Set maximum number of tribs
                                  41  A5    17  90  01A7  1149          MOVB    #DT$_DMV11,UCB$B_DEVTYPE(R5)    ; Set device type to be DMV11
                                    012C  C5  D5  01AB  1150          TSTL    UCB$L_XD_UV1BUF(R5)             ; Is the buffer area allocated?
                                          7C  12  01AF  1151          BNEQ    INIT_OTHERS                     ; Br if yes, continue
                                          52  D4  01B1  1152          CLRL    R2                              ; Zero end marker
                                          52  DD  01B3  1153 10$:      PUSHL   R2                              ; Save bad buffer on stack
                                0000180C  8F  D0  01B5  1154 20$:      MOVL    #UV1_BUFFER_AREA+12,R1          ; Compute size of buffer area
                                              01BC  1155                                                      ; ..that must be contiguous
                                00000000'GF   16  01BC  1156          JSB     G^EXE$ALONONPAGED               ; Allocate buffer
```

```
                 5B 50   E9  01C2  1157          BLBC    R0,50$                          ; Br if none available
                             01C5  1158          ASSUME  IRP$W_SIZE LT 12
                             01C5  1159          ASSUME  IRP$B_TYPE EQ IRP$W_SIZE+2
          0013180C 8F    D0  01C5  1160          MOVL    #<DYN$C_BUFIO@16>.<UV1_BUFFER_AREA+12>,-; Set size and type
                   08 A2       01CB  1161                  IRP$W_SIZE(R2)                  ; of buffer
                               01CD  1162          ;
                               01CD  1163          ; Make sure buffer area is contiguous:
                               01CD  1164          ;
           50    0C A2    9E  01CD  1165          MOVAB   12(R2),R0                       ; Skip header
50    FFFFFE00 8F    CA  01D1  1166          BICL    #-512,R0                        ; Compute BOFF of buffer area
50    000019FF 8F    C0  01D8  1167          ADDL    #UV1_BUFFER_AREA+511,R0         ; Add in length of area that
                               01DF  1168                                          ;  must be contiguous & roundup
50    000001FF 8F    CA  01DF  1169          BICL    #511,R0                         ; Round to page boundry
 50     50    F7 8F    78  01E6  1170          ASHL    #-VA$V_VPN,R0,R0                ; Compute number of pages-1
           51    0C A2    9E  01EB  1171          MOVAB   12(R2),R1                       ; Get address of buffer area
                       09   EF  01EF  1172          EXTZV   S^#VA$V_VPN,-                   ; Get system page table number
      54    51          15       01F1  1173                  S^#VA$S_VPN,R1,R4               ;
53    00000000'GF       D0  01F4  1174          MOVL    G^MMG$GL_SPTBASE,R3             ; Get system page table base
        54    6344      DE  01FB  1175          MOVAL   (R3)[R4],R4                     ; Compute PTE address
                       00   EF  01FF  1176          EXTZV   S^#PTE$V_PFN,-                  ; Get page frame number of
      53    64          15       0201  1177                  S^#PTE$S_PFN,(R4),R3            ;  first page
           54    04   C0  0204  1178          ADDL    #4,R4                           ; Get address of next PTE
                       0D   11  0207  1179          BRB     40$                             ; Get into loop
                               0209  1180 30$:
                       00   EF  0209  1181          EXTZV   S^#PTE$V_PFN,-                  ; Get page frame number of
      51    64          15       020B  1182                  S^#PTE$S_PFN,(R4),R1            ;  next page
           54    04   C0  020E  1183          ADDL    #4,R4                           ; Get address of next PTE
           51    53   D1  0211  1184          CMPL    R3,R1                           ; Is buffer contiguous?
                       9D   12  0214  1185          BNEQ    10$                             ; Br if not. try again
                               0216  1186 40$:
                       53   D6  0216  1187          INCL    R3                              ; Compute next page frame number
          EE 50    F5  0218  1188          SOBGTR  R0,30$                          ; Loop if more pages
012C C5    52    D0  021B  1189          MOVL    R2,UCB$L_XD_UV1BUF(R5)          ; Save buffer area address
                               0220  1190
           50 8ED0      0220  1191 50$:    POPL    R0                              ; Get next buffer address
           08   13      0223  1192          BEQL    60$                             ; Br if no more
 00000000'GF    16      0225  1193          JSB     G^COM$DRVDEALMEM                ; Deallocate the bad buffers
           F3   11      022B  1194          BRB     50$                             ; Look for more
                               022D  1195 60$:
                               022D  1196
                               022D  1197 INIT_OTHERS:
15 64 A5   05    E1      022D  1198          BBC     #UCB$V_POWER,UCB$W_STS(R5),90$  ; Branch if not powerfail
           00    E1      0232  1199          BBC     #UCB$V_XD_INITED,-              ; Br if not inited
     10 68 A5             0234  1200                  UCB$W_DEVSTS(R5),90$            ;
01 A3    40 8F    90      0237  1201          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R3)      ; Master clear the device
 53    01   15    78      023C  1202          ASHL    #XD_BSEL2_V_ERR+16,#1,R3        ; Indicate error
 54    01   10    78      0240  1203          ASHL    #16,#1,R4                       ; Indicate powerfail
          165C    30      0244  1204          BSBW    SCHED_FORK                      ; Schedule fork process
           1F    BA      0247  1205 90$:    POPR    #^M<R0,R1,R2,R3,R4>             ; Restore registers
                 05      0249  1206          RSB                                     ; Return
```

```
                                024A  1208          .SBTTL  XMT_FDT,  Transmit I/O Operation FDT Routine
                                024A  1209
                                024A  1210 ;++
                                024A  1211 ; XMT_FDT - Transmit I/O Operation FDT Routine
                                024A  1212 ;
                                024A  1213 ; Functional description:
                                024A  1214 ;
                                024A  1215 ; This routine is called by the SYS$QIO system service to dispatch a
                                024A  1216 ; WRITE I/O request.
                                024A  1217 ;
                                024A  1218 ; The QIO parameters for WRITEs are:
                                024A  1219 ;
                                024A  1220 ;       P1 = address of the buffer
                                024A  1221 ;       P2 = size of the buffer
                                024A  1222 ;       All other parameters are unused.
                                024A  1223 ;
                                024A  1224 ; The buffer is validated for access and locked into memory. The I/O operation
                                024A  1225 ; is started if possible else it is blocked temporarily on one of the various
                                024A  1226 ; wait queues of the UCB or CDB.
                                024A  1227 ;
                                024A  1228 ;
                                024A  1229 ; Inputs:
                                024A  1230 ;
                                024A  1231 ;       R3 = IRP address (I/O request packet)
                                024A  1232 ;       R4 = PCB address (process control block)
                                024A  1233 ;       R5 = UCB address (unit control block)
                                024A  1234 ;       R6 = CCB address (channel control block)
                                024A  1235 ;       R7 = bit number of the I/O function code
                                024A  1236 ;
                                024A  1237 ; Outputs:
                                024A  1238 ;
                                024A  1239 ;       R0 = status of transmit request initiation
                                024A  1240 ;
                                024A  1241 ;       R1,R2,R8,R9 are destroyed
                                024A  1242 ;       R3-R7 are preserved.
                                024A  1243 ;
                                024A  1244 ;--
                                024A  1245
                                024A  1246 XMT_FDT:                                     ; Transmit FDT routine
          40 A3     7C          024A  1247          CLRQ    IRP$Q_STATION(R3)         ; Clear trib address field
        50   14     3C          024D  1248          MOVZWL  S^#SS$_BADPARAM,R0        ; Assume bad parameters
        58   6C     D0          0250  1249          MOVL    P1(AP),R8                 ; Get starting address of user buffer
     59   04 AC     3C          0253  1250          MOVZWL  P2(AP),R9                 ; Get length of user buffer
           7C     13            0257  1251          BEQL    ABORTIO                   ; Br if zero length buffer
        50   58     7D          0259  1252          MOVQ    R8,R0                     ; Retrieve buffer parameters
     00000000'GF     16          025C  1253          JSB     G^EXE$WRITECHK            ; Check accessibility of user buffer
                                 0262  1254                                            ; (No return on NO ACCESS)
                                 0262  1255                                            ; Returns IRP$W_BCNT
  51 00000048 8F     C0          0262  1256          ADDL2   #CXB$C_HEADER,R1          ; Calculate length of buffer needed
           38     BB            0269  1257          PUSHR   #^M<R3,R4,R5>             ; Save registers
     00000000'GF     16          026B  1258          JSB     G^EXE$BUFFRQUOTA          ; Check if process has sufficient quota
           5F 50     E9          0271  1259          BLBC    R0,20$                    ; Br if quota check failure
     00000000'GF     16          0274  1260          JSB     G^EXE$ALLOCBUF            ; Allocate CXB buffer for output
           56 50     F9          027A  1261          BLBC    R0,20$                    ; If LBC allocation failure
        53   6E     D0          027D  1262          MOVL    (SP),R3                   ; Retrieve address of IRP
     50   0080 C4     D0          0280  1263          MOVL    PCB$L_JIB(R4),R0          ; Get JIB address
        20 A0   51     C2          0285  1264          SUBL    R1,JIB$L_BYTCNT(R0)       ; Adjust buffered I/O quota
```

```
                30 A3    51    B0  0289  1265          MOVW    R1,IRP$W_BOFF(R3)       ; Set number of bytes charged to quota
                2C A3    52    D0  028D  1266          MOVL    R2,IRP$L_SVAPTE(R3)     ; Save CXB address in IRP
                                   0291  1267
                                   0291  1268          ASSUME  CXB$L_FL EQ 0
                                   0291  1269          ASSUME  CXB$L_BL EQ CXB$L_FL+4
                82    48 A2    9E  0291  1270          MOVAB   CXB$C_HEADER(R2),(R2)+  ; Set pointer to start of data
                      82    D4  0295  1271          CLRL    (R2)+                   ; Clear next link cell
                                   0297  1272
                                   0297  1273          ASSUME  CXB$W_SIZE EQ CXB$L_BL+4
                   82    51    B0  0297  1274          MOVW    R1,(R2)+                ; Set size of structure
                                   029A  1275
                                   029A  1276          ASSUME  CXB$B_TYPE EQ CXB$W_SIZE+2
                                   029A  1277          ASSUME  CXB$B_CODE EQ CXB$B_TYPE+1
                   82    1B    9B  029A  1278          MOVZBW  #DYN$C_CXB,(R2)+        ; Set structure type
                                   029D  1279
                                   029D  1280          ASSUME  CXB$W_LENGTH EQ CXB$B_CODE+1
                   52    3C    C0  029D  1281          ADDL    #CXB$C_HEADER-CXB$W_LENGTH,R2; Get address of data portion of buffer
             62    68    59    28  02A0  1282          MOVC3   R9,(R8),(R2)            ; Move data to system buffer
                         38    BA  02A4  1283          POPR    #^M<R3,R4,R5>           ; Restore registers
                                   02A6  1284
                      1C45    30  02A6  1285          BSBW    XLATE                   ; Get CDB address
                   29 50      E9  02A9  1286          BLBC    R0,ABORTIO              ; Br if error
                                   02AC  1287          SETIPL  UCB$B_FIPL(R5)          ; Sync acces to UCB
                                   02B0  1288
                                   02B0  1290          PUSHQ   R6                      ; Save registers
                   50    01    90  02B3  1291          MOVB    #JNL_XMT,R0             ; Set journal type = XMIT
                      2214    30  02B6  1292          BSBW    XD_FILL_JNX             ; Fill the journal buffer
                   04 50      E9  02B9  1293          BLBC    R0,10$                  ; Br if error
             86    32 A3    B0  02BC  1294          MOVW    IRP$W_BCNT(R3),(R6)+    ; Store byte count
                                   02C0  1295 10$:     POPQ    R6                      ; Restore registers
                                   02C3  1297
                         54    DD  02C3  1298          PUSHL   R4                      ; Save R4
                         20    10  02C5  1299          BSBB    XMT_START               ; Start transmit operation
                         54  8ED0  02C7  1300          POPL    R4                      ; Restore R4
                      08 50    E9  02CA  1301          BLBC    R0,ABORTIO              ; Br if error
          00000000'GF    17  02CD  1302          JMP     G^EXE$QIORETURN         ; Exit QIO service to await completion
                                   02D3  1303
                         38    BA  02D3  1304 20$:     POPR    #^M<R3,R4,R5>           ; Restore registers
                                   02D5  1305
                                   02D5  1306 ABORTIO:                             ; Abort the I/O request
          00000000'GF    17  02D5  1307          JMP     G^EXE$ABORTIO           ; and exit QIO service
```

```
                                                .SBTTL  XMT_START,  Start Transmit Operation
                                        ;++
                                        ; XMT_START - Start Transmit Operation
                                        ;
                                        ; Functional description:
                                        ;
                                        ; This routine is called to start a transmit operation. If the tributary
                                        ; is running and there are not too many transmit requests already pending,
                                        ; then the request is given to the device to transmit immediately if possible.
                                        ; If there are no more mapping register slots available, or a UNIBUS mapping
                                        ; register set cannot be allocated then the request is queued to the UCB xmit
                                        ; wait queue.
                                        ;
                                        ; If the circuit is not running yet or the circuit has too many request
                                        ; pending, then the request is queued to the CDB xmit wait queue.
                                        ;
                                        ; Inputs:
                                        ;
                                        ;       R3 = IRP address
                                        ;       R5 = UCB address
                                        ;       R9 = CDB address
                                        ;
                                        ;       IPL = FIPL
                                        ;
                                        ; Outputs
                                        ;
                                        ;       R0 = status of transmit request
                                        ;
                                        ;       R3-R7 are preserved.
                                        ;
                                        ;--
                                                .ENABL  LSB
                                        TRIB_INACT:
    50    20D4 8F    3C                 MOVZWL  #SS$_DEVINACT,R0                ; Trib is not active
                34    11                BRB     15$                             ; Return error
                                        TOO_BIG:
          50    14    3C                MOVZWL  #SS$_BADPARAM,R0                ; Buffer is too large
                2F    11                BRB     15$                             ; Return error
                                        XMT_START:                              ; Start transmit operation
                0B    E1                BBC     #XM$V_STS_ACTIVE,-              ; Br if trib not active
                  F0 69                         CDB_L_DEVDEPEND(R9),TRIB_INACT  ;
  FE5E CF     32 A3    B1                CMPW    IRP$W_BCNT(R3),MAX_W_BUFFER     ; Is buffer too large?
                EF    1A                BGTRU   TOO_BIG                         ; Br if buffer is too big
                                        ;
                                        ; Insert entry on CDB transmit wait queue if circuit not running or
                                        ; there are too many transmit requests pending.
                                        ;
      21 A3     11    90                MOVB    #XD_FC_V_XMIT,IRP$B_XDFUNC(R3)  ; Set function request in IRP
      14 B9     63    0E                INSQUE  (R3),@CDB_Q_XMT_REQ+4(R9)       ; Insert IRP on CDB xmit wait Q
                                        XMT_ALT_START:                          ; Alternate start of xmits
                                        ;
                                        ; Inputs:
                                        ;       R9 = CDB address
```

```
                                02FB  1366 ;
                          0D    E1  02FB  1367          BBC     #XM$V_STS_RUNNING,-              ; Br if trib is not running
                        1D 69       02FD  1368                  CDB_L_DEVDEPEND(R9),30$          ;
                     53   10 B9     02FF  1369          REMQUE  @CDB_Q_XMT_REQ(R9),R3            ; Get oldest xmit request
                          17  1D     0303  1370          BVS     30$                              ; Br if none - process others
                          0F  A9  97  0305  1371          DECB    CDB_B_XMT_CNT(R9)                ; One less xmit possible for this ci
                          0D  18      0308  1372          BGEQ    20$                              ; Br if we can handle this request
                          0F  A9  96  030A  1373          INCB    CDB_B_XMT_CNT(R9)                ; Else, set count back to zero
                     10 A9    63  0E  030D  1374          INSQUE  (R3),CDB_Q_XMT_REQ(R9)           ; Re-insert on front of queue
                              09  11  0311  1375          BRB     30$                              ; Try to start waiting xmits
                                      0313  1376
                        50    01  9A  0313  1377 10$:     MOVZBL  S^#SS$_NORMAL,R0                 ; Return success
                              05  00  0316  1378 15$:     RSB
                                      0317  1379
                   00A0 D5    63  0E  0317  1380 20$:     INSQUE  (R3),@UCB$Q_XD_XMT_REQ+4(R5)     ; Insert request on UCB wait Q
                                      031C  1381 ;
                                      031C  1382 ; Find a free mapping register slot.  If none currently available, put the
                                      031C  1383 ; I/O request in the CCB wait queue.
                                      031C  1384 ;
                          05  E0  031C  1385 30$:     BBS     #UCB$V_XD_FORK_XMT,-             ; Br if XMT fork block in use
                     F2 68 A5        031E  1386                  UCB$W_DEVSTS(R5),10$             ;
                  54   0133 C5  9A  0321  1387          MOVZBL  UCB$B_XD_XMT_MAX(R5),R4          ; Get max concurrent transmits
54   0131 C5       54    00  EB  0326  1388          FFC     #0,R4,UCB$B_XD_XMT_MAP(R5),R4    ; Find a free transmit slot
                              E4  13  032D  1389          BEQL    10$                              ; Br if none free
                                      032F  1390 ;
                                      032F  1391 ; Store IRP info in UCB
                                      032F  1392 ;
                  53   009C D5    0F  032F  1393          REMQUE  @UCB$Q_XD_XMT_REQ(R5),R3         ; Get oldest xmit request
                              DD  1D  0334  1394          BVS     10$                              ; Br if none, leave
                                      0336  1395
                                      0336  1396          SETBIT  R4,UCB$B_XD_XMT_MAP(R5)          ; Set mapping slot in use flag
                     3C A3    54  90  033C  1397          MOVB    R4,IRP$L_MEDIA+4(R3)             ; Save mapping slot number used
                                      0340  1398 ;
                                      0340  1399 ; Skip MAP REGISTER useage if u-VAX I.
                                      0340  1400 ;
                                      0340  1401          CPUDISP <<790,XMT_OTHERS>,-
                                      0340  1402                   <780,XMT_OTHERS>,-
                                      0340  1403                   <750,XMT_OTHERS>,-
                                      0340  1404                   <730,XMT_OTHERS>,-
                                      0340  1405                   <UV1,XMT_UV1>>
                                      035A  1406
                                      035A  1407 XMT_UV1:
                  3A A3    32 A3    B0  035A  1408          MOVW    IRP$W_BCNT(R3),IRP$L_MEDIA+2(R3); Save BCNT in device format
                50   010C C544     D0  035F  1409          MOVL    UCB$L_XD_XMT_PA(R5)[R4],R0       ; Get buffer Physical address
                     38 A3    50  B0  0365  1410          MOVW    R0,IRP$L_MEDIA(R3)               ; Store BA00-BA15
             50      50    F0 8F  78  0369  1411          ASHL    #-16,R0,R0                       ; Shift down high byte
                     3D A3    50  90  036E  1412          MOVB    R0,IRP$L_MEDIA+5(R3)             ; Store BA16-BA21
                                      0372  1413          ;
                                      0372  1414          ; Copy the data to the contiguous buffer.
                                      0372  1415          ;
                51   0124 C544     D0  0372  1416          MOVL    UCB$L_XD_XMT_VA(R5)[R4],R1       ; Get contiguous buffer's VA
                     52   2C B3    D0  0378  1417          MOVL    @IRP$[_SVAPTE(R3),R2             ; Get system buffer's VA
                             38    BB  037C  1418          PUSHR   #^M<R3,R4,R5>
             61      62    32 A3    28  037E  1419          MOVC3   IRP$W_BCNT(R3),(R2),(R1)         ; Do the copy
                             38    BA  0383  1420          POPR    #^M<R3,R4,R5>
                          0076     31  0385  1421          BRW     90$                              ; Continue
                                      0388  1422
```

```
                                0388  1423 XMT_OTHERS:
      7E A5    32 A3    B0    0388  1424          MOVW    IRP$W_BCNT(R3),UCB$W_BCNT(R5)   ; Set buffer count
         50    2C B3    D0    038D  1425          MOVL    @IRP$L_SVAPTE(R3),R0            ; Get buffer virtual address
            FE00 8F    AB    0391  1426          BICW3   #^C<VA$M_BYTE>,-                ; Compute BOFF
         7C A5    50          0395  1427                   R0,UCB$W_BOFF(R5)               ;
50    50    15    09    EF    0398  1428          EXTZV   #VA$V_VPN,#VA$S_VPN,R0,R0       ; Get virtual page number
51     00000000'GF    D0    039D  1429          MOVL    G^MMG$GL_SPTBASE,R1             ; Get the base address of SPTEs
      78 A5    6140    DE    03A4  1430          MOVAL   (R1)[R0],UCB$L_SVAPTE(R5)       ; Set address of the SPTE
                                03A9  1431 ;
                                03A9  1432 ; Allocate UNIBUS map registers
                                03A9  1433 ;
            20    A8    03A9  1434          BISW    #UCB$M_XD_FORK_XMT,-            ; Assume we will have to wait
         68 A5          03AB  1435                  UCB$W_DEVSTS(R5)                ; ..for fork to run
      000003BB'EF    9F    03AD  1436          PUSHAB  40$                             ; Push address of fork
      00000000'GF    17    03B3  1437          JMP     G^IOC$REQMAPREG                 ; Request some map registers
                                03B9  1438 ;
                                03B9  1439 ; The following code may be executed as a fork process, therefore we have
                                03B9  1440 ; to setup a timeout service routine offset.
                                03B9  1441 ;
                          20CB' 03B9  1442          .WORD   TIMEOUT-.                       ; Timeout service routine
            20    AA    03BB  1443 40$:    BICW    #UCB$M_XD_FORK_XMT,-            ; Fork block is no longer in
         68 A5          03BD  1444                  UCB$W_DEVSTS(R5)                ; ..use
            00    E0    03BF  1445          BBS     #CD_TS_V_ESTAB,-                ; Br if trib STILL established
      0C 0C A9          03C1  1446                  CDB_W_STS(R9),50$               ;
                        03C4  1447          RELMPR                                  ; Else, release map registers
         50    2C    3C    03CA  1448          MOVZWL  #SS$_ABORT,R0                   ; Return request in error
            1800    31    03CD  1449          BRW     IO_DONE                         ; Complete the I/O request
                                03D0  1450
            57    DD    03D0  1451 50$:    PUSHL   R7                              ; Save R7
      57    24 A5    D0    03D2  1452          MOVL    UCB$L_CRB(R5),R7                ; Get CRB address
                                03D6  1453          ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                                03D6  1454          ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
            34 A7    D0    03D6  1455          MOVL    CRB$L_INTD+VEC$W_MAPREG(R7),-   ; Save mapping info
         00E0 C544          03D9  1456                  UCB$L_XD_XMT_MAP(R5)[R4]        ;
                                03DD  1457 ;
                                03DD  1458 ; Map the buffer
                                03DD  1459 ;
                                03DD  1460          ASSUME  UCB$W_BOFF+2 EQ UCB$W_BCNT
      38 A3    7C A5    D0    03DD  1461          MOVL    UCB$W_BOFF(R5),IRP$L_MEDIA(R3)  ; Move byte offset and size
            34 A7    F0    03E2  1462          INSV    CRB$L_INTD+VEC$W_MAPREG(R7),-   ; Insert BA9-BA15
   38 A3    07    09          03E5  1463                  #9,#7,IRP$L_MEDIA(R3)
50 34 A7    06    07    EF    03E9  1464          EXTZV   #7,#6,CRB$L_INTD+VEC$W_MAPREG(R7),R0 ; Get BA16-BA17
38 A3    02    1E    50    F0    03EF  1465          INSV    R0,#30,#2,IRP$L_MEDIA(R3)       ; Insert BA16-BA17
      00000000'GF    16    03F5  1466          JSB     G^IOC$LOADUBAMAPA               ; Loadmap registers
            57 8EDC    03FB  1467          POPL    R7                              ; Restore R7
                                03FE  1468
                                03FE  1469 90$:
                                03FE  1470 ;
                                03FE  1471 ; Request and load the port with the buffer address and size, and return
                                03FE  1472 ;
      0098 D5    63    0E    03FE  1473          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)     ; Insert transmit request at end of
                                0403  1474                                                  ;     UCB input wait queue.
            59    DD    0403  1475          PUSHL   R9                              ; Save CDB address
                        0405  1476          DSBINT  UCB$B_DIPL(R5)                  ; Sync access to device
               0BC1    30    040C  1477          BSBW    LOAD_PORT                       ; Load port
                        040F  1478          ENBINT                                  ; Restore IPL
            59 8ED0    0412  1479          POPL    R9                              ; Restore CDB address
```

```
FEE3   31  0415  1480          BRW     XMT_ALT_START                   ; Lets try it again
           0418  1481          .DSABL  LSB
```

```
                                0418  1483          .SBTTL  RCV_FDT,  Read I/O Operation FDT Routine
                                0418  1484
                                0418  1485 ;++
                                0418  1486 ; RCV_FDT - Read I/O Operation FDT Routine
                                0418  1487 ;
                                0418  1488 ; Functional description:
                                0418  1489 ;
                                0418  1490 ; This routine is called by the SYS$QIO system service to dispatch a
                                0418  1491 ; READ I/O request.
                                0418  1492 ;
                                0418  1493 ; The QIO parameters for READs are:
                                0418  1494 ;
                                0418  1495 ;       P1 = address of the buffer
                                0418  1496 ;       P2 = size of the buffer
                                0418  1497 ;       All other parameters are unused.
                                0418  1498 ;
                                0418  1499 ; The specified buffer is checked for accessibility. The buffer address and
                                0418  1500 ; count are saved in the packet. Then IPL is raised to device fork IPL and if
                                0418  1501 ; a message is available the operation is complete. Otherwise the packet is
                                0418  1502 ; queued onto the waiting receive list of the CDB.
                                0418  1503 ;
                                0418  1504 ; If the function specifies a modifier of IO$M_NOW, if no message is
                                0418  1505 ; available when the test is made, a status of SS$_ENDOFFILE is
                                0418  1506 ; returned.
                                0418  1507 ;
                                0418  1508 ; Inputs:
                                0418  1509 ;
                                0418  1510 ;       R3 = I/O Packet Address
                                0418  1511 ;       R4 = PCB Address
                                0418  1512 ;       R5 = UCB Address
                                0418  1513 ;       R6 = CCB Address
                                0418  1514 ;       R7 = Function Code
                                0418  1515 ;       AP = Address of first operation-specific QIO parameter
                                0418  1516 ;
                                0418  1517 ; Outputs:
                                0418  1518 ;
                                0418  1519 ;       R0 = status of the receive QIO operation
                                0418  1520 ;       R3-R7 - unchanged
                                0418  1521 ;
                                0418  1522 ;--
                                0418  1523
                                0418  1524 RCV_FDT:                                    ; Read FDT process routine
      40 A3    7C               0418  1525          CLRQ    IRP$Q_STATION(R3)         ; Clear tributary address field
                                041B  1526 ;
                                041B  1527 ; Check request parameters
                                041B  1528 ;
      50    14 3C               041B  1529          MOVZWL  S^#SS$_BADPARAM,R0        ; Assume bad parameters
   51    04 AC 3C               041E  1530          MOVZWL  P2(AP),R1                 ; Get length of buffer
         31    13               0422  1531          BEQL    10$                       ; Br if zero - abort I/O
      50    0C D0               0424  1532          MOVL    P1(AP),R0                 ; Get buffer address
   38 A3    50 D0               0427  1533          MOVL    R0,IRP$L_MEDIA(R3)        ; Save buffer addr in IRP
      30 A3    B4               042B  1534          CLRW    IRP$W_BOFF(R3)            ; No quota to return during completi
00000000'GF    16               042E  1535          JSB     G^EXE$READCHK             ; Check accessibility
                                0434  1536                                            ; (No return on no access)
       1AB7    30               0434  1537          BSBW    XLATE                     ; Get CDB address
      1B 50    E9               0437  1538          BLBC    R0,10$                    ; Br if error
                                043A  1539          SETIPL  UCB$B_FIPL(R5)            ; Raise IPL to fork level
```

```
                     043E  1540
                     043E  1542          PUSHQ   R6                      ;  to lock the data base
                     043E  1542                                          ; Save registers
50   02      90     0441  1543          MOVB    #JNL_RCV,R0             ; Set journal type = RECV
   2086      30     0444  1544          BSBW    XD_FILL_JNX             ; Fill the journal buffer
                     0447  1545          POPQ    R6                      ; Restore registers
                     044A  1547
     0C      10     044A  1548          BSBB    RCV_START               ; Start receive operation
  06 50      E9     044C  1549          BLBC    R0,10$                  ; Br if error
00000000'GF  17     044F  1550          JMP     G^EXE$QIORETURN         ; Return to await completion
                     0455  1551
   FE7D      31     0455  1552 10$:     BRW     ABORTIO                 ; Abort the I/O request
```

```
                     0458  1554          .SBTTL  RCV_START,  Start Receive Operation
                     0458  1555
                     0458  1556 ;++
                     0458  1557 ; RCV_START - Start Receive Operation
                     0458  1558 ;
                     0458  1559 ; Functional description:
                     0458  1560 ;
                     0458  1561 ; At this point, the buffer is okay and we are now at fork IPL.  Now check
                     0458  1562 ; to make sure the tributary has been established.  If not, we will return
                     0458  1563 ; an SS$_DEVOFFLINE error status.  Note that the circuit need not be turned
                     0458  1564 ; on in order to do a read request.
                     0458  1565 ;
                     0458  1566 ; Inputs:
                     0458  1567 ;
                     0458  1568 ;       R3 = IRP address
                     0458  1569 ;       R5 = UCB address
                     0458  1570 ;       R9 = CDB address
                     0458  1571 ;
                     0458  1572 ;       IPL = FIPL
                     0458  1573 ;
                     0458  1574 ; Outputs:
                     0458  1575 ;
                     0458  1576 ;       R0 = return status of receive request
                     0458  1577 ;       R3-R7 are preserved.
                     0458  1578 ;
                     0458  1579 ;--
                     0458  1580          .ENABL  LSB
                     0458  1581 RCV_START:
 50   20D4 8F    3C  0458  1582          MOVZWL  #SS$_DEVINACT,R0                ; Assume trib not active
            0B    E1  045D  1583          BBC     #XM$V_STS_ACTIVE,-              ; Br if trib not active
          4C 69        045F  1584                  CDB_L_DEVDEPEND(R9),40$         ;
   0C A9    04    E1  0461  1585          BBC     #CD_TS_V_BUFQUO,CDB_W_STS(R9),- ; Br if no buffer management
            26        0465  1586                  RCV_START_ALT                   ;
   21 A3    12    90  0466  1587          MOVB    #XD_FC_V_RECV,IRP$B_XDFUNC(R3)  ; Set function request in IRP
09 20 A3    06    E1  046A  1588          BBC     #IO$V_NOQ,IRP$W_FUNC(R3),5$     ; Br if not READ NOW
   52    20 A9    9E  046F  1589          MOVAB   CDB_Q_RCV_MSG(R9),R2            ; Get address of received msgs
      52    62    D1  0473  1590          CMPL    (R2),R2                         ; Is there a message?
            24    13  0476  1591          BEQL    15$                             ; Br if no - return EOF
                      0478  1592 5$:       DSBINT  UCB$B_DIPL(R5)                  ; Sync access to device
0098 D5    63    0E  047F  1593          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)     ; Insert at end of input Q
        0B49    30  0484  1594          BSBW    LOAD_PORT                       ; Bump quota for trib
                      0487  1595          ENBINT                                  ; Restore IPL
            1E    11  048A  1596          BRB     30$                             ; Return with success
                      048C  1597
                      048C  1598 RCV_START_ALT:                                  ; Alternate receive startup
                      048C  1599 ;
                      048C  1600 ;  Check to see if message is available
                      048C  1601 ;
   52    20 B9    0F  048C  1602          REMQUE  @CDB_Q_RCV_MSG(R9),R2           ; Dequeue a received message
            05    1D  0490  1603          BVS     10$                             ; Br if none
                      0492  1604 ;
                      0492  1605 ; Complete the receive with available message
                      0492  1606 ;
          16FE    30  0492  1607          BSBW    FINISH_RCV_IO                   ; Complete the I/O request
            13    11  0495  1608          BRB     30$                             ; and exit
                      0497  1609 ;
                      0497  1610 ; No message available.  If IO$M_NOW specified, return with SS$_ENDOFFILE
```

```
                        0497  1611 ; status. Otherwise, queue IRP to await arrival of message and continue.
                        0497  1612 ;
0A 20 A3    06    E1    0497  1613 10$:    BBC     #IO$V_NOW,IRP$W_FUNC(R3),20$    ; Branch if not READ NOW
 50   0870 8F    3C    049C  1614 15$:    MOVZWL  #SS$_ENDOFFILE,R0               ; Set no message status
         172C    30    04A1  1615         BSBW    IO_DONE                         ; Complete the I/O
           04    11    04A4  1616         BRB     30$                             ; and exit
                        04A6  1617 ;
                        04A6  1618 ; Queue the IRP to CDB receive IRP wait queue
                        04A6  1619 ;
   1C B9    63    0E    04A6  1620 20$:    INSQUE  (R3),@CDB_Q_RCV_REQ+4(R9)       ; Queue IRP to await message
      50    01    9A    04AA  1621 30$:    MOVZBL  S^#SS$_NORMAL,R0                ; Set successful return
            05    05    04AD  1622 40$:    RSB                                     ; Return
                        04AE  1623         .DSABL  LSB
```

```
                     04AE  1625          .SBTTL  ALT_START,  Alternate Start I/O Routine
                     04AE  1626
                     04AE  1627 ;++
                     04AE  1628 ; ALT_START - Alternate Start I/O Routine
                     04AE  1629 ;
                     04AE  1630 ; Functional description:
                     04AE  1631 ;
                     04AE  1632 ; This entry point is used by the "internal IRP" interface.  IRP's
                     04AE  1633 ; which enter the driver at this point have not been pre-processed
                     04AE  1634 ; by our FDT routines.  These IRP's have been built by another
                     04AE  1635 ; driver and look exactly like a regular IRP.
                     04AE  1636 ;
                     04AE  1637 ; This routine, therefore, must do the same processing which the FDT
                     04AE  1638 ; routines would do if they had been called as in the case of a normal
                     04AE  1639 ; IRP.
                     04AE  1640 ;
                     04AE  1641 ; NOTE: The PID and CHAN fields of the IRP are sufficient to map to a
                     04AE  1642 ; CDB.
                     04AE  1643 ;
                     04AE  1644 ; Inputs:
                     04AE  1645 ;
                     04AE  1646 ;       R3 - IRP address
                     04AE  1647 ;       R5 - UCB address
                     04AE  1648 ;
                     04AE  1649 ;       All pertinent fields of the IRP are assumed to be valid.
                     04AE  1650 ;
                     04AE  1651 ;       IPL = FIPL
                     04AE  1652 ;
                     04AE  1653 ; Outputs:
                     04AE  1654 ;
                     04AE  1655 ;       R0 = status of the transmit/receive request.
                     04AE  1656 ;
                     04AE  1657 ;       R3-R5 are preserved.
                     04AE  1658 ;
                     04AE  1659 ;--
                     04AE  1660
                     04AE  1661 ALT_START:
          59   DD    04AE  1662          PUSHL   R9                              ; Save R9
        40 A3  7C    04B0  1663          CLRQ    IRP$Q_STATION(R3)               ; Clear station field
        28 A3  B5    04B3  1664          TSTW    IRP$W_CHAN(R3)                  ;****temporary****
           05  19    04B6  1665          BLSS    3$                              ;****temporary****
28 A3   28 A3  AE    04B8  1666          MNEGW   IRP$W_CHAN(R3),IRP$W_CHAN(R3)   ;****temporary****
 50   02A0 C5  D0    04BD  1667 3$:      MOVL    UCB$L_XD_PID(R5),R0             ; Use starter's PID
         1A2D  30    04C2  1668          BSBW    XLATE_ALT                       ; Get CDB address
        38 50  E9    04C5  1669          BLBC    R0,30$                          ; Br if error
18 2A A3   01  E0    04C8  1670          BBS     #IRP$V_FUNC,IRP$W_STS(R3),10$   ; Br if read request
                     04CD  1671
                     04CD  1673          PUSHQ   R6                              ; Save registers
        50   01  90    04D0  1674          MOVB    #JNL_XMT,R0                     ; Set journal type = XMIT
          1FF7  30    04D3  1675          BSBW    XD_FILL_JNX                     ; Fill the journal buffer
        04 50  E9    04D6  1676          BLBC    R0,5$                           ; Br if error
    86  32 A3  B0    04D9  1677          MOVW    IRP$W_BCNT(R3),(R6)+            ; Store byte count
                     04DD  1678 5$:      POPQ    R6                              ; Restore registers
                     04E0  1680
          FE04  30    04E0  1681          BSBW    XMT_START                       ; Initiate the transmit
            1B  11    04E3  1682          BRB     30$                             ; Return
                     04E5  1683
```

```
                                                H  5
                     04E5  1684 10$:
                     04E5  1686          PUSHQ   R6                              ; Save registers
          50   02    90   04E8  1687          MOVB    #JNL_RCV,R0                 ; Set journal type = RECV
            1FDF    30   04EB  1688          BSBW    XD_FILL_JNX                 ; Fill the journal buffer
                     04EE  1689          POPQ    R6                              ; Restore registers
                     04F1  1691
     52   2C A3    D0   04F1  1692          MOVL    IRP$L_SVAPTE(R3),R2         ; Get address of input buffer
               06    13   04F5  1693          BEQL    20$                         ; Br if none
          2C A3    D4   04F7  1694          CLRL    IRP$L_SVAPTE(R3)            ; Make sure SVAPTE is cleared
            09E3    30   04FA  1695          BSBW    ADDRCVLIST                  ; Else, add it to the receive list
            FF58    30   04FD  1696 20$:     BSBW    RCV_START                   ; Initiate the read
               59 8ED0   0500  1697 30$:     POPL    R9                          ; Restore R9
          01 50    E9   0503  1698          BLBC    R0,40$                      ; Branch if error
                    05   0506  1699          RSB                                 ; Return to caller
                     0507  1700
            16C6    31   0507  1701 40$:     BRW     IO_DONE                     ; Post the I/O request in error
```

```
                        .SBTTL  SETMODE_FDT,  Set mode I/O operation FDT routine

                ;++
                ; SETMODE_FDT - Set mode I/O operation FDT routine
                ;
                ; Functional description:
                ;
                ; This routine is used to set the configuration of the DMP11 hardware
                ; device.  Subfunction modifier bits are used to specify the type of
                ; action to be taken.  The two characteristics buffers (P1 and P2)
                ; are used to describe specific characteristics.
                ;
                ; The QIO parameters for SETMODE are:
                ;
                ;       P1 = Optional address of quadword or longword buffer
                ;       P2 = Optional address of buffer descriptor for extended characteristics
                ;       P3 = Number of receive buffers to pre-allocate. Required on
                ;            controller startup.
                ;
                ;
                ; The subfunction modifiers are as follows:
                ;
                ;
                ;        o STARTUP - establish a tributary - this modifier is used to
                ;              establish a tributary so that communication between
                ;              a unit and the DMP11 device can occur.  Polling
                ;              parameters may also be modified with this command
                ;              either at startup time or at any subsequent point
                ;              in time.
                ;
                ;       o SHUTDOWN - shutdown the assigned tributary - this modifier
                ;              is used to stop the polling and delete the tributary
                ;              by the user of that tributary.  Only the assigned
                ;              tributary may be shutdown and this command does not
                ;              require privilege.
                ;
                ;       o ATTNAST - request an attention AST - this modifier is used
                ;              to set up an AST to be delivered when a change of
                ;              status occurs on this tributary.
                ;
                ;       o CTRL - perform the request on the Controller not the
                ;              tributary.
                ;
                ;       o SET_MODEM - set line unit's mode register.
                ;
                ; Inputs:
                ;
                ;       R3 = IRP address
                ;       R4 = PCB address
                ;       R5 = UCB address
                ;       R6 = CCB address
                ;       R7 = Function code
                ;       AP = address of first QIO parameter
                ;
                ; Outputs:
                ;
                ;       R0 = status of setmode request
```

```
                     050A  1760 ;
                     050A  1761 ;       R3-R5 are preserved.
                     050A  1762 ;
                     050A  1763 ;       R7-R9 = destroyed
                     050A  1764 ;
                     050A  1765 ;--
                     050A  1766
                     050A  1767 SETMODE_FDT:                                 ; Setmode FDT processing
        40 A3   7C   050A  1768         CLRQ    IRP$Q_STATION(R3)           ; Clean up station field
                     050D  1769
                     050D  1771         PUSHQ   R6                          ; Save registers
    50   03   90   0510  1772         MOVB    #JNL_QIO,R0                 ; Set journal type = QIO
      1FB7   30   0513  1773         BSBW    XD_FILL_JNX                 ; Fill the journal buffer
      04 50   E9   0516  1774         BLBC    R0,1$                       ; Br if error
F8 A6   6C   D0   0519  1775         MOVL    P1(AP),-8(R6)               ; Store the P1 parameter
                     051D  1776 1$:     POPQ    R6                          ; Restore registers
                     0520  1778
57   20 A3   3C   0520  1779         MOVZWL  IRP$W_FUNC(R3),R7           ; Get entire function code
04 57   0A   E0   0524  1780         BBS     #IO$V_SET_MODEM,R7,3$       ; Br if SET_MODEM request
03 57   09   E1   0528  1781         BBC     #IO$V_CTRL,R7,5$            ; Br if not controller request
      020A   31   052C  1782 3$:      BRW     SETMODE_CTRL                ; Process controller request
                     052F  1783 ;
                     052F  1784 ; Perform setmode request on a tributary
                     052F  1785 ;
36 57   08   E1   052F  1786 5$:      BBC     #IO$V_ATTNAST,R7,30$        ; Branch if not attention AST
                     0533  1787
                     0533  1788 ;
                     0533  1789 ; User is requesting an attention AST.
                     0533  1790 ;
      19B8   30   0533  1791         BSBW    XLATE                       ; Get CDB address
    33 50   E9   0536  1792         BLBC    R0,49$                      ; Br if error - abort I/O
  57   04 A9   DE   0539  1793         MOVAL   CDB_L_AST(R9),R7            ; Get addr of AST list
00000000'GF   16   053D  1794         JSB     G^COM$SETATTNAST            ; Set up attention AST
11 0C A9   00   E1   0543  1795         BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),20$ ; Br if tributary not estab
  51   20 A9   9E   0548  1796         MOVAB   CDB_Q_RCV_MSG(R9),R1        ; Check for empty rcv list
    61   51   D1   054C  1797         CMPL    R1,(R1)                     ; Empty?
        08   13   054F  1798         BEQL    20$                         ; Yes, no need to inform user
        53   DD   0551  1799 10$:     PUSHL   R3                          ; Else, save IRP address
      1D1A   30   0553  1800         BSBW    POKE_USER                   ; Inform the user
        53 8ED0   0556  1801         POPL    R3                          ; Restore IRP address
    51   69   D0   0559  1802 20$:     MOVL    CDB_L_DEVDEPEND(R9),R1      ; Get CDB dependent status
  51   44 A5   C8   055C  1803         BISL    UCB$L_DEVDEPEND(R5),R1      ; OR in UCB status
    50   01   3C   0560  1804 23$:     MOVZWL  S^#SS$_NORMAL,R0            ; Set success
00000000'GF   17   0563  1805 25$:     JMP     G^EXE$FINISHIO              ; Complete the I/O
                     0569  1806
                     0569  1807
      02FE   30   0569  1808 30$:     BSBW    GET_CHAR_BUFS               ; Get P1 and P2 characteristics
    30 50   E9   056C  1809 49$:     BLBC    R0,40$                      ; Br if error - abort I/O
 2F 57   07   E1   056F  1810         BBC     #IO$V_SHUTDOWN,R7,50$       ; Branch if not trib shutdown
                     0573  1811 ;
                     0573  1812 ; Shutdown tributary modifier specified.
                     0573  1813 ;
                     0573  1814 ; Validate P2 buffer. Try to find CDB, if found then update CDB and stop
                     0573  1815 ; tributary.
                     0573  1816 ;
      1978   30   0573  1817         BSBW    XLATE                       ; Get CDB address
        52   D4   0576  1818         CLRL    R2                          ; Assume no CDB present
```

```
                04 50    E9  0578  1819          BLBC    R0,33$                                  ; Br if not CDB present
           52   0C A9    3C  057B  1820          MOVZWL  CDB_W_STS(R9),R2                        ; Else, get status
                 1B07    30  057F  1821 33$:     BSBW    VALIDATE_P2_CDB                         ; Validate the P2 buffer
                DE 50    E9  0582  1822          BLBC    R0,25$                                  ; Br if error
           51   44 A5    D0  0585  1823          MOVL    UCB$L_DEVDEPEND(R5),R1                  ; Assume no CDB
                   59    D5  0589  1824          TSTL    R9                                      ; Was CDB found?
                   D3    13  058B  1825          BEQL    23$                                     ; No - leave now
                 1AA9    30  058D  1826          BSBW    CHG_CDB                                 ; Change CDB parameters
        C4 0C A9   00    E1  0590  1827          BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),20$ ; Br if nothing to delete
           21 A3   02    90  0595  1828          MOVB    S^#XD_FC_V_STOPT,IRP$B_XDFUNC(R3) ; Set internal function code
                 0522    31  0599  1829          BRW     QUEPKT                                  ; Queue packet to driver
                         059C  1830
                   53 8ED0  059C  1831 37$:     POPL    R3                                      ; Restore R3
                 FD33    31  059F  1832 40$:     BRW     ABORTIO                                 ; Abort the I/O request
                         05A2  1833 ;
                         05A2  1834 ; Startup tributary modifier specified or no modifier
                         05A2  1835 ;
                 1949    30  05A2  1836 50$:     BSBW    XLATE                                   ; Get CDB address
                03 50    E9  05A5  1837          BLBC    R0,55$                                  ; Br if CDB address not found
                 0166    31  05A8  1838          BRW     150$                                    ; Else, change existing trib
                         05AB  1839 ;
                         05AB  1840 ; We are creating a new tributary.
                         05AB  1841 ;
           50    20D4 8F    3C  05AB  1842 55$:     MOVZWL  #SS$_DEVINACT,R0                        ; Assume ctrl not active
                   00    E1  05B0  1843          BBC     #UCB$V_XD_INITED,-                      ; Br if ctrl not inited
                EA 68 A5       05B2  1844                  UCB$W_DEVSTS(R5),40$                    ;
                         05B5  1845 ;
                         05B5  1846 ; Allow only 1 trib for point to point or tributary mode
                         05B5  1847 ;
        0E 68 A5   02    E1  05B5  1848          BBC     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),60$ ; Br if not point to point
                         05BA  1849 ;&&&         BBS     #XM$V_CHR_CTRL,-                        ; Br if control station
                         05BA  1850 ;&&&                 UCB$L_DEVDEPEND(R5),60$                 ;
        02A0 C5    0C A3    D1  05BA  1851          CMPL    IRP$L_PID(R3),UCB$L_XD_PID(R5)  ; Make sure same PID
                      0C    12  05C0  1852          BNEQ    70$                                     ; Br if not - sneaky devil
                0134 C5     95  05C2  1853          TSTB    UCB$B_XD_TRB_CNT(R5)                    ; Any tribs yet?
                      06    12  05C6  1854          BNEQ    70$                                     ; Br if trib already exists
                         05C8  1855 ;
                         05C8  1856 ; Find an empty tributary slot in UCB
                         05C8  1857 ;
                    18CC 30  05C8  1858 60$:     BSBW    FIND_SLOT                               ; Find an empty slot
                   07 50 E8  05CB  1859          BLBS    R0,80$                                  ; Br if found one
           50    0850 8F    3C  05CE  1860 70$:     MOVZWL  #SS$_DEVICEFULL,R0                      ; Return error
                      CA    11  05D3  1861          BRB     40$                                     ; Abort I/O
                         05D5  1862 ;
                         05D5  1863 ; Found empty slot in UCB translation vector, validate P2 and allocate a CDB
                         05D5  1864 ;
                      52 D4  05D5  1865 80$:     CLRL    R2                                      ; No status flags yet
                      59 D4  05D7  1866          CLRL    R9                                      ; No CDB address - Yet
                    1AAD 30  05D9  1867          BSBW    VALIDATE_P2_CDB                         ; Validate the P2 buffer
                   19 50 E9  05DC  1868          BLBC    R0,83$                                  ; Br if error
                         05DF  1869 ;
                         05DF  1870 ; Make sure trib address given if startup request.
                         05DF  1871 ; We make this requirement, because this is a new CDB and the tributary
                         05DF  1872 ; address is required for all modes except Point to Point.
                         05DF  1873 ;
              18 57    06  E1  05DF  1874          BBC     #IO$V_STARTUP,R7,85$                    ; Br if not startup request
           13 68 A5    02  E0  05E3  1875          BBS     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),85$ ; Br if point-to-point
```

```
                51   0474 8F    3C  05E8  1876          MOVZWL  #NMA$C_PCCI_TRI,R1                ; Get trib address
                       1C53    30  05ED  1877          BSBW    UNPACK_P2_BUF                   ; From P2 buffer
                     08 50     E8  05F0  1878          BLBS    R0,85$                          ; Br if given - Okay
                50   0114 8F    3C  05F3  1879          MOVZWL  #SS$_INSFARG,R0                 ; Insufficient arguments
                       FF68    31  05F8  1880 83$:     BRW     25$                             ; Finish the I/O request
                               05FB  1881
                51   0060 8F    3C  05FB  1882 85$:     MOVZWL  #CDB_C_LENGTH,R1                ; Get size of CDB
                          53    DD  0600  1883          PUSHL   R3                              ; Save IRP address
                00000000'GF     16  0602  1884          JSB     G^EXE$BUFQUOPRC                 ; Check caller's quota
                        91 50   E9  0608  1885          BLBC    R0,37$                          ; Br if error
                00000000'GF     16  060B  1886          JSB     G^EXE$ALONONPAGED               ; Allocate the CDB
                        88 50   E9  0611  1887          BLBC    R0,37$                          ; Br if error
                          53 8ED0   0614  1888          POPL    R3                              ; Restore IRP address
                50   0080 C4    D0  0617  1889          MOVL    PCB$L_JIB(R4),R0                ; Get JIB address
          20 A0  00000060 8F    C2  061C  1890          SUBL    #CDB_C_LENGTH,JIB$L_BYTCNT(R0)  ; Charge user for CDB
          24 A0  00000060 8F    C2  0624  1891          SUBL    #CDB_C_LENGTH,JIB$L_BYTLM(R0)   ; ..and limit
                               062C  1892 ;
                               062C  1893 ; Zero the CDB
                               062C  1894 ;
                      59   52  D0  062C  1895          MOVL    R2,R9                           ; Copy CDB address
                         38   BB  062F  1896          PUSHR   #^M<R3,R4,R5>                   ; Save registers
   62   0060 8F    00   62   00  2C  0631  1897          MOVC5   #0,(R2),#0,#CDB_C_LENGTH,(R2)   ; Zero the CDB
                         38   BA  0639  1898          POPR    #^M<R3,R4,R5>                   ; Restore registers
              48 A9   0C A3   D0  063B  1899          MOVL    IRP$L_PID(R3),CDB_L_PID(R9)     ; Save starter's PID
              4C A9   28 A3   B0  0640  1900          MOVW    IRP$W_CHAN(R3),CDB_W_CHAN(R9)   ; Save associated channel
                               0645  1901          ASSUME  CDB_B_TYPE EQ CDB_W_SIZE+2
                               0645  1902          ASSUME  CDB_B_POL EQ CDB_B_TYPE+1
                               0645  1903 ;
                               0645  1904 ; We will init the Polling state to unlatched. It is assumed that
                               0645  1905 ; a zero polling state means unlatched!
                               0645  1906 ;
             00130060 8F    D0  0645  1907          MOVL    #<<DYN$C_BUFIO@16>!CDB_C_LENGTH>,- ; Set type and size
                   08 A9        064B  1908                  CDB_W_SIZE(R9)                  ;
                               064D  1909 ;
                               064D  1910 ; Initialize CDB queues
                               064D  1911 ;
                50    05   9A  064D  1912          MOVZBL  S^#CDB_C_QUEUES,R0              ; Set number of queue heads
             52   10 A9   9E  0650  1913          MOVAB   CDB_Q_QUEUES(R9),R2             ; Set address of first head
                82    62   9E  0654  1914 90$:     MOVAB   (R2),(R2)+                      ; Init forward link pointer
                82   FC A2   9E  0657  1915          MOVAB   -4(R2),(R2)+                    ; Init backward link pointer
                   F6 50    F5  065B  1916          SOBGTR  R0,90$                          ; Loop if more queues
                               065E  1917 ;
                               065E  1918 ; Initialize CDB default parameters
                               065E  1919 ;
                50    12   3C  065E  1920          MOVZWL  #DEF_TRIB_PARAMSZ,R0            ; Set size of parameters in bytes
             51   FAC1 CF    9E  0661  1921          MOVAB   DEF_TRIB_PARAM,R1               ; Set address of defaults
               52   4E A9   9E  0666  1922          MOVAB   CDB_B_SETPRM(R9),R2             ; Set address of parameters
                81    81   90  066A  1923 100$:     MOVB    (R1)+,(R2)+                     ; Set default
                   FA 50    F5  066D  1924          SOBGTR  R0,100$                         ; Loop for all parameters
                               0670  1925
          5C A9   02B4 C5    B0  0670  1926          MOVW    UCB$W_XD_RTT(R5),CDB_W_SEL_TIMER(R9) ; Set default retransmit timer
          0F A9   0135 C5    90  0676  1927          MOVB    UCB$B_XD_XMT_TRB(R5),CDB_B_XMT_CNT(R9) ; Init max xmits allowed
            05 44 A5    06   E0  067C  1928          BBS     #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5),110$ ; Br if control mode
            0F A9    7F 8F   90  0681  1929          MOVB    #INF_XMT,CDB_B_XMT_CNT(R9)      ; Else, allow infinite xmits
               58    41 A3   9A  0686  1930 110$:     MOVZBL  IRP$B_INDEX(R3),R8              ; Get vector index
                               068A  1931          SETIPL  UCB$B_FIPL(R5)                  ; Sync access to UCB
           0218 C548    59   D0  068E  1932          MOVL    R9,UCB$L_XD_CDB_VEC(R5)[R8]     ; Store CDB address in UCB
```

```
                44 A3   0218 C548   DE  0694  1933          MOVAL   UCB$L_XD_CDB_VEC(R5)[R8],-      ; Store address of CDB address
                                    0698  1934                  IRP$L_CDB(R3)                   ; ..in CDB
                        0C A3       D0  0698  1935          MOVL    IRP$L_PID(R3),-                 ; Save PID in UCB
                     0158 C548          069E  1936                  UCB$L_XD_PID_VEC(R5)[R8]        ;
                        28 A3       B0  06A2  1937          MOVW    IRP$W_CHAN(R3),-                ; Save channel index in UCB
                     01D8 C548          06A5  1938                  UCB$W_XD_CHAN_VEC(R5)[R8]       ;
                       0134 C5      96  06A9  1939          INCB    UCB$B_XD_TRB_CNT(R5)            ; Tally one more tributary
                                    06AD  1940 ;
                                    06AD  1941 ; Tributary now exists change its characteristics and set them
                                    06AD  1942 ; if trib is established.
                                    06AD  1943 ;
                            1989    30  06AD  1944 120$:   BSBW    CHG_CDB                         ; Change CDB parameters
                          10 57 06  E0  06B0  1945          BBS     #IO$V_STARTUP,(R7),125$         ; Br if startup request
                             50 01  3C  06B4  1946          MOVZWL  #SS$_NORMAL,R0                  ; Else, set successful return
                             51 69  D0  06B7  1947          MOVL    CDB_L_DEVDEPEND(R9),R1          ; Set IOSB1 return
                           51 44 A5 C8  06BA  1948          BISL    UCB$L_DEVDEPEND(R5),R1          ; ..plus UCB status
                     00000000'GF    17  06BE  1949 123$:   JMP     G^EXE$FINISHIO                  ; Finish the I/O request
                                    06C4  1950 ;
                                    06C4  1951 ; Startup tributary requested
                                    06C4  1952 ;
                          21 A3  00 90  06C4  1953 125$:   MOVB    S^#XD_FC_V_INITT,IRP$B_XDFUNC(R3) ; Set internal function code
                     02A0 C5  48 A9 D1  06C8  1954          CMPL    CDB_L_PID(R9),UCB$L_XD_PID(R5)  ; Same PID as starter?
                                09 12  06CE  1955          BNEQ    127$                            ; Br if not
                             09 69  00 E1  06D0  1956          BBC     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),128$ ; Br if not MOP mode
                          04 44 A5  06 E1  06D4  1957          BBC     #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5),128$ ; Br if not control station
                             4E A9  01 90  06D9  1958 127$:   MOVB    #1,CDB_B_MRB(R9)                ; Else, set MRB to one
                          4E A9  FF 8F  91  06DD  1959 128$:   CMPB    #INF_RCV,CDB_B_MRB(R9)          ; Is MRB at 'unlimited'?
                                05 13  06E2  1960          BEQL    129$                            ; Br if yes
                                    06E4  1961          SETBIT  #CD_TS_V_BUFQUO,CDB_W_STS(R9)   ; Else, must user buffer manager
                       12 68 A5  02 E1  06E9  1962 129$:   BBC     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),130$ ; Br if not point-point
                          0E A9  01 90  06EE  1963          MOVB    #1,CDB_B_TRB_ADDR(R9)           ; Else set trib address
                          40 A3  01 90  06F2  1964          MOVB    #1,IRP$Q_STATION(R3)            ; Again in IRP
                          50  41 A3 9A  06F6  1965          MOVZBL  IRP$B_INDEX(R3),R0              ; Get vector index
                     0138 C540   01 90  06FA  1966          MOVB    #1,UCB$L_XD_TRIB_VEC(R5)[R0]    ; And again in UCB trib vector
                              0E A9 95  0700  1967 130$:   TSTB    CDB_B_TRB_ADDR(R9)              ; Trib address given?
                                31 12  0703  1968          BNEQ    170$                            ; Br if yes - okay
                       50   0114 8F 3C  0705  1969          MOVZWL  #SS$_INSFARG,R0                 ; Trib address missing
                       51   0474 8F 3C  070A  1970          MOVZWL  #NMA$C_PCCI_TRI,R1              ; Set bad parameter code
                                AD 11  070F  1971          BRB     123$                            ; Finish the I/O
                                    0711  1972 ;
                                    0711  1973 ; Tributary already exists .. validate P2
                                    0711  1974 ;
                             52  69 D0  0711  1975 150$:   MOVL    CDB_L_DEVDEPEND(R9),R2          ; Get trib status
                                1972 30  0714  1976          BSBW    VALIDATE_P2_CDB                 ; Validate P2
                                  A4 50  E9  0717  1977          BLBC    R0,123$                         ; Br if error
                                    071A  1978          SETIPL  UCB$B_FIPL(R5)                  ; Sync access to CDB
                     04 0C A9   00 E0  071E  1979          BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),160$ ; Br if trib established
                                    0723  1980 ;
                                    0723  1981 ; Tributary not active .. change CDB parameters and do startup if necessary
                                    0723  1982 ;
                                69  94  0723  1983          CLRB    CDB_L_DEVDEPEND(R9)             ; Clear old characteristics
                                86  11  0725  1984          BRB     120$                            ; Change CDB and start if needed
                                    0727  1985 ;
                                    0727  1986 ; Tributary already active .. set CDB and give request to device
                                    0727  1987 ;
                              3C A3 94  0727  1988 160$:   CLRB    IRP$L_MEDIA+4(R3)               ; Clear trib characteristics
                               190C 30  072A  1989          BSBW    CHG_CDB                         ; Change the CDB params
```

```
                21 A3   05    90  072D  1990          MOVB    S^#XD_FC_V_CHGT,IRP$B_XDFUNC(R3) ; Set new CDB params
             38 A3   FF 8F    90  0731  1991          MOVB    #255,IRP$L_MEDIA(R3)             ; Set 8 parameters
                     0385    31  0736  1992 170$:     BRW     QUEPKT                           ; Queue packet to driver
```

```
                                0739  1994          .SBTTL  SETMODE_CTRL,  Perform setmode FDT operation on controller
                                0739  1995
                                0739  1996 ;++
                                0739  1997 ; SETMODE_CTRL - Perform setmode FDT operation on controller
                                0739  1998 ;
                                0739  1999 ; Functional description:
                                0739  2000 ;
                                0739  2001 ; This routine performs the SETMODE FDT setup for the controller.
                                0739  2002 ;
                                0739  2003 ; Inputs:
                                0739  2004 ;
                                0739  2005 ;       R3 = IRP address
                                0739  2006 ;       R4 = PCB address
                                0739  2007 ;       R5 = UCB address
                                0739  2008 ;       R7 = IRP function word
                                0739  2009 ;
                                0739  2010 ; Outputs:
                                0739  2011 ;
                                0739  2012 ;       R0 = status of setmode request
                                0739  2013 ;
                                0739  2014 ;       R3-R5 are preserved.
                                0739  2015 ;
                                0739  2016 ;--
                                0739  2017
                                0739  2018 SETMODE_CTRL:                                   ; Perform setmode on controller
1C 57   0A      E1              0739  2019          BBC     #IO$V_SET_MODEM,R7,10$          ; Br if not setmodem request
                                073D  2020 ;
                                073D  2021 ; Set modem modifier specified
                                073D  2022 ;
                                073D  2023 ;       P1(AP) = address of modem longword bit settings
                                073D  2024 ;
   50   14   3C                  073D  2025          MOVZWL  #SS$_BADPARAM,R0                ; Assume P1 buffer not given
   52   6C   D0                  0740  2026          MOVL    P1(AP),R2                       ; Get address of P1 buffer
        44   13                  0743  2027          BEQL    20$                             ; Br if none specified
   50   0C   3C                  0745  2028          MOVZWL  #SS$_ACCVIO,R0                  ; Assume access violation
                                0748  2029          IFNORD  #4,(R2),20$                     ; Br if access violation
38 A3   62   D0                  074E  2030          MOVL    (R2),IRP$L_MEDIA(R3)            ; Store modem bits in IRP
21 A3   0D   90                  0752  2031          MOVB    S^#XD_FC_V_WTMODEM,IRP$B_XDFUNC(R3) ; Set function request
      0365   31                  0756  2032          BRW     QUEPKT                          ; Queue packet to driver
                                0759  2033
      010E   30                 0759  2034 10$:      BSBW    GET_CHAR_BUFS                   ; Get P1 and P2 characteristics
     2A 50   E9                 075C  2035          BLBC    R0,20$                          ; Br if error - Abort I/O
52   68 A5   3C                 075F  2036          MOVZWL  UCB$W_DEVSTS(R5),R2             ; Get device status
      192F   30                 0763  2037          BSBW    VALIDATE_P2_UCB                 ; Validate the P2 buffer
     1A 50   E9                 0766  2038          BLBC    R0,17$                          ; Br if error
                                0769  2039
1F 57   07      E1              0769  2040          BBC     #IO$V_SHUTDOWN,R7,25$           ; Br if not shutdown request
                                076D  2041 ;
                                076D  2042 ; Shutdown modifier specified
                                076D  2043 ;
      17F6   30                 076D  2044          BSBW    CHG_UCB                         ; Update the UCB
        00   E1                 0770  2045          BBC     #UCB$V_XD_INITED,-              ; Br if controller not up
  07 68 A5                      0772  2046                  UCB$W_DEVSTS(R5),15$            ;
21 A3   03   90                 0775  2047          MOVB    #XD_FC_V_STOPC,IRP$B_XDFUNC(R3) ; Set function request
      0342   31                 0779  2048          BRW     QUEPKT                          ; Queue packet to driver
                                077C  2049
51   44 A5   D0                 077C  2050 15$:     MOVL    UCB$L_DEVDEPEND(R5),R1          ; Get IOSB1 return
```

```
                50    01    9A  0780  2051          MOVZBL  S^#SS$_NORMAL,R0                ; Set success
      00000000'GF       17  0783  2052 17$:        JMP     G^EXE$FINISHIO                  ; Complete the I/O request
                        0789  2053
            FB49        31  0789  2054 20$:        BRW     ABORTIO                         ; Abort the I/O request
                        078C  2055 ;
                        078C  2056 ; Make sure this request is from the starter PID unless device not inited
                        078C  2057 ; or device is running in control station mode.
                        078C  2058 ;
                  00    E1  078C  2059 25$:        BBC     #UCB$V_XD_INITED,-               ; Br if device not inited
               12 68    A5  078E  2060                     UCB$W_DEVSTS(R5),30$            ;
                  06    E0  0791  2061          BBS     #XM$V_CHR_CTRL,-                 ; Br if control station
               0D 44    A5  0793  2062                     UCB$L_DEVDEPEND(R5),30$         ;
            50  0850    8F  3C  0796  2063          MOVZWL  #SS$_DEVICEFULL,R0               ; Assume someone has device
      02A0 C5    0C    A3  D1  079B  2064          CMPL    IRP$L_PID(R3),UCB$L_XD_PID(R5)   ; Is this the owner PID?
                  E6    12  07A1  2065          BNEQ    20$                              ; Br if no - wrong user
                        07A3  2066
               03 57    06  E0  07A3  2067 30$:        BBS     #IO$V_STARTUP,R7,31$             ; Br if startup request
                     007B   31  07A7  2068          BRW     50$                              ; Else, no modifier
                        07AA  2069 ;
                        07AA  2070 ; Controller startup requested
                        07AA  2071 ;
                        07AA  2072 ;       P3(AP) = number of receive buffers desired
                        07AA  2073 ;
               51    08 AC    3C  07AA  2074 31$:        MOVZWL  P3(AP),R1                ; Get number of receive buffers
                     1B       13  07AE  2075          BEQL    35$                      ; Br if no - ignore it
                  00    E1  07B0  2076          BBC     #UCB$V_XD_INITED,-               ; Br if device not inited
               11 68    A5  07B2  2077                     UCB$W_DEVSTS(R5),33$            ;
            02AB C5    51  91  07B5  2078          CMPB    R1,UCB$B_XD_BFN(R5)              ; Are the buffer numbers the same?
                  0A       13  07BA  2079          BEQL    33$                      ; Br if yes
            51  0451    8F  3C  07BC  2080 32$:        MOVZWL  #NMA$C_PCLI_BFN,R1       ; Set IOSB1 return
                  50    14  3C  07C1  2081          MOVZWL  #SS$_BADPARAM,R0         ; Set error return
                        BD  11  07C4  2082          BRB     17$                      ; Finish the I/O request
            02AB C5    51  90  07C6  2083 33$:        MOVB    R1,UCB$B_XD_BFN(R5)              ; Store new receive buffer number
                        07CB  2084 ;
                        07CB  2085 ; Set new P1, P2 parameters
                        07CB  2086 ;
                  00    E0  07CB  2087 35$:        BBS     #UCB$V_XD_INITED,-               ; Br if device already inited
               74 68    A5  07CD  2088                     UCB$W_DEVSTS(R5),70$            ;
               03 38    A3  E9  07D0  2089          BLBC    IRP$L_MEDIA(R3),37$              ; Br if no P1
                  44    A5  94  07D4  2090          CLRB    UCB$L_DEVDEPEND(R5)              ; Clear old characteristics
                  178C      30  07D7  2091 37$:        BSBW    CHG_UCB                          ; Change the UCB characteristics
            51    02AB C5    9A  07DA  2092          MOVZBL  UCB$B_XD_BFN(R5),R1              ; Get number of receive buffers
              52    3A    A3  3C  07DF  2093          MOVZWL  IRP$L_MEDIA+2(R3),R2             ; Get message size from P1
               04 38    A3  E8  07E3  2094          BLBS    IRP$L_MEDIA(R3),40$              ; Br if P1 buffer valid
              52    42    A5  3C  07E7  2095          MOVZWL  UCB$W_DEVBUFSIZ(R5),R2           ; Else, get buffer size from UCB
                  50    14  3C  07EB  2096 40$:        MOVZWL  #SS$_BADPARAM,R0         ; Assume bad parameter
                  51    52  C4  07EE  2097          MULL    R2,R1                    ; Compute total needed for buffers
                        C9  13  07F1  2098          BEQL    32$                      ; Br if zero - error
                  57    51  3C  07F3  2099          MOVZWL  R1,R7                    ; Copy quota
                  57    >1  D1  07F6  2100          CMPL    R1,R7                    ; Overflow?
                        C1  12  07F9  2101          BNEQ    32$                      ; Br if error
                        53  DD  07FB  2102          PUSHL   R3                       ; Save R3
             00000000'GF       16  07FD  2103          JSB     G^EXE$BUFQUOPRC          ; Check caller's quota
                        53  8ED0  0803  2104          POPL    R3                       ; Restore R3
                     B3 50    E9  0806  2105          BLBC    R0,32$                   ; Br if error
              42 A3    57  B0  0809  2106          MOVW    R7,IRP$W_QUOTA(R3)       ; Save quota in packet
            50    0080 C4    D0  080D  2107          MOVL    PCB$L_JIB(R4),R0         ; Get JIB address
```

```
                20 A0    57   C2  0812  2108          SUBL    R7,JIB$L_BYTCNT(R0)                 ; Charge user for rec. bufs
                24 A0    57   C2  0816  2109          SUBL    R7,JIB$L_BYTLM(R0)                  ; ..and limit
                21 A3    01   90  081A  2110          MOVB    S^#XD_FC_V_INITC,IRP$B_XDFUNC(R3) ; Set function request in IRP
                39 A3    01   90  081E  2111          MOVB    S^#XD_FC_V_INITC,IRP$L_MEDIA+1(R3) ; ..Twice for fork process
                      0299   31  0822  2112          BRW     QUEPKT                              ; Queue request to driver
                                 0825  2113 ;
                                 0825  2114 ; No modifier specified - change controller parameters
                                 0825  2115 ;
                         00     E0  0825  2116 50$:    BBS     #UCB$V_XD_INITED,-                  ; Br if already inited
                   1A 68 A5             0827  2117                  UCB$W_DEVSTS(R5),70$          ;
                   03 38 A3     E9  082A  2118          BLBC    IRP$L_MEDIA(R3),53$                 ; Br if no P1 buffer
                      44 A5     94  082E  2119          CLRB    UCB$L_DEVDEPEND(R5)                 ; Clear old UCB characteristics
                       1732     30  0831  2120 53$:    BSBW    CHG_UCB                             ; Change UCB parameters
                     50   01     3C  0834  2121          MOVZWL  S^#SS$_NORMAL,R0                    ; Else, set success
                  51    44 A5     D0  0837  2122          MOVL    UCB$L_DEVDEPEND(R5),R1              ; Set IOSB1 return
               00000000'GF       17  083B  2123 55$:    JMP     G^EXE$FINISHIO                      ; Finish the I/O request
                                 0841  2124
                       FA91      31  0841  2125 60$:    BRW     ABORTIO                             ; Abort the I/O request
                                 0844  2126 ;
                                 0844  2127 ; Device already inited - set new parameters and give them to device
                                 0844  2128 ;
                   0F 38 A3     E9  0844  2129 70$:    BLBC    IRP$L_MEDIA(R3),80$                 ; Br if no P1 buffer
                44 A5    3C A3     91  0848  2130          CMPB    IRP$L_MEDIA+4(R3),UCB$L_DEVDEPEND(R5) ; Are characteristics okay?
                            08     13  084D  2131          BEQL    80$                                 ; Yes - let it go
                         50   14     3C  084F  2132          MOVZWL  #SS$_BADPARAM,R0                    ; Return error
                         51   01     CE  0852  2133          MNEGL   S^#1,R1                             ; No specific parameter
                              E4     11  0855  2134          BRB     55$                                 ; Complete the I/O
                                     0857  2135
                       170C      30  0857  2136 80$:    BSBW    CHG_UCB                             ; Change UCB parameters
                   21 A3    04     90  085A  2137          MOVB    S^#XD_FC_V_CHGC,IRP$B_XDFUNC(R3) ; Set function request
                   39 A3    04     90  085E  2138          MOVB    S^#XD_FC_V_CHGC,IRP$L_MEDIA+1(R3) ; ..Twice for fork process
               38 A3    F0 8F     90  0862  2139          MOVB    #<15@4>,IRP$L_MEDIA(R3)             ; Set only four parameters
                       0254      31  0867  2140          BRW     QUEPKT                              ; Queue packet to driver
```

```
                        086A  2142          .SBTTL  GET_CHAR_BUFS,  Get P1 and P2 characteristics buffers
                        086A  2143
                        086A  2144 ;++
                        086A  2145 ; GET_CHAR_BUFS - Get P1 and P2 characteristics buffers
                        086A  2146 ;
                        086A  2147 ; Functional description:
                        086A  2148 ;
                        086A  2149 ; This routine saves the P1 and P2 buffers for later use by the driver.
                        086A  2150 ; The P1 buffer is saved in the IRP (IRP$L_MEDIA) as a quadword value.
                        086A  2151 ; The P2 buffer is saved by allocating the appropriate amount of memory from
                        086A  2152 ; non-paged pool.   The user's quota is checked before the allocation is made.
                        086A  2153 ; And the non-paged pool buffer is charged against the user's quota. The P2
                        086A  2154 ; system buffer address is passed in IRP$L_SVAPTE of the IRP.
                        086A  2155 ;
                        086A  2156 ;
                        086A  2157 ; Inputs:
                        086A  2158 ;
                        086A  2159 ;       R3 = IRP address
                        086A  2160 ;       R4 = PCB address
                        086A  2161 ;       R5 = UCB address
                        086A  2162 ;
                        086A  2163 ; Outputs:
                        086A  2164 ;
                        086A  2165 ;       R0 = status of buffers
                        086A  2166 ;
                        086A  2167 ;       R3-R5 are preserved.
                        086A  2168 ;
                        086A  2169 ;--
                        086A  2170
                        086A  2171 GET_CHAR_BUFS:                                  ; Get characteristics buffers
                        086A  2172 ;
                        086A  2173 ; Check access to P1 buffer and save P1 characteristics
                        086A  2174 ;
      38 A3    7C       086A  2175          CLRQ    IRP$L_MEDIA(R3)               ; Reset P1 chars
   52    6C    D0       086D  2176          MOVL    P1(AP),R2                     ; Get address of P1 char buf
         11    13       0870  2177          BEQL    10$                           ; Branch if no P1 buffer
   50    0C    3C       0872  2178          MOVZWL  #SS$_ACCVIO,R0                ; Assume access violation
                        0875  2179          IFNORD  #8,(R2),20$                   ; Check access
38 A3    62    7D       0878  2180          MOVQ    (R2),IRP$L_MEDIA(R3)          ; Save P1 characteristics
38 A3    01    90       087F  2181          MOVB    #1,IRP$L_MEDIA(R3)            ; Indicate valid P1 buffer
                        0883  2182 ;
                        0883  2183 ; Check access to P2 buffer and check process's buffer quota
                        0883  2184 ;
51    04 AC    D0       0883  2185 10$:     MOVL    P2(AP),R1                     ; Get address P2 char buf desc
         33    13       0887  2186          BEQL    40$                           ; Br if no P2 buffer
         53    DD       0889  2187          PUSHL   R3                            ; Save R3
00000000'GF    16       088B  2188          JSB     G^EXE$PROBER_DSC              ; Check access to buffer
      0E 50    E9       0891  2189          BLBC    R0,15$                        ; Br if error
   51    51    3C       0894  2190          MOVZWL  R1,R1                         ; Get buffer size as a word
         52    DD       0897  2191          PUSHL   R2                            ; Save R2
00000000'GF    16       0899  2192          JSB     G^EXE$BUFQUOPRC               ; Check for buffered quota
         52 8ED0        089F  2193          POPL    R2                            ; Restore R2
         53 8ED0        08A2  2194 15$:     POPL    R3                            ; Restore R3
      01 50    E8       08A5  2195          BLBS    R0,30$                        ; Branch if quota ok
               05       08A8  2196 20$:     RSB                                   ; Return
                        08A9  2197
                        08A9  2198 ;
```

```
                     08A9  2199 ; Quota OKAY, allocate buffer and copy info.
                     08A9  2200 ;
          0261   30  08A9  2201 30$:    BSBW    ALLOC_P2BUF                     ; Allocate buffer
        10 50    E9  08AC  2202         BLBC    R0,50$                          ; Br if error
     50   2C A3   D0  08AF  2203         MOVL    IRP$L_SVAPTE(R3),R0             ; Get P2 buffer address
             38   BB  08B3  2204         PUSHR   #^M<R3,R4,R5>                   ; Save sacred registers
 0C A0   62   51   28  08B5  2205         MOVC3   R1,(R2),P2B_T_DATA(R0)          ; Save P2 char buffer
             38   BA  08BA  2206         POPR    #^M<R3,R4,R5>                   ; Restore registers
        50   01   9A  08BC  2207 40$:    MOVZBL  S^#SS$_NORMAL,R0                ; Set success
                  05  08BF  2208 50$:    RSB                                     ; Return
```

```
                        08C0  2210          .SBTTL  SENSEMODE_FDT,  Sense Mode I/O operation FDT routine
                        08C0  2211
                        08C0  2212 ;++
                        08C0  2213 ; SENSEMODE_FDT - Sense Mode FDT routine
                        08C0  2214 ;
                        08C0  2215 ; Functional Description:
                        08C0  2216 ;
                        08C0  2217 ;  This routine returns information to the caller about the configuration
                        08C0  2218 ;  and status of the DMP11 device.  Depending on the function modifier,
                        08C0  2219 ;  either the device characteristics, error counters or modem register
                        08C0  2220 ;  contents are returned.
                        08C0  2221 ;
                        08C0  2222 ;
                        08C0  2223 ;  The QIO parameters for SENSEMODE are:
                        08C0  2224 ;
                        08C0  2225 ;       P1 = optional address of quadword or longword buffer
                        08C0  2226 ;       P2 = optional address of buffer descriptor for extended characteristics
                        08C0  2227 ;
                        08C0  2228 ;
                        08C0  2229 ; Inputs:
                        08C0  2230 ;
                        08C0  2231 ;       R3 = IRP address
                        08C0  2232 ;       R4 = PCB address
                        08C0  2233 ;       R5 = UCB address
                        08C0  2234 ;       R6 = CCB address
                        08C0  2235 ;       R7 = Function code
                        08C0  2236 ;       AP = Address of first function-dependent QIO parameter
                        08C0  2237 ;
                        08C0  2238 ; Outputs:
                        08C0  2239 ;
                        08C0  2240 ;       R0 = status return of sensemode request
                        08C0  2241 ;
                        08C0  2242 ;       R3-R5 are preserved.
                        08C0  2243 ;
                        08C0  2244 ;--
                        08C0  2245
                        08C0  2246 SENSEMODE_FDT:                                  ; Sensemode FDT I/O processing
    40 A3       7C      08C0  2247          CLRQ    IRP$Q_STATION(R3)              ; Clear station address field
                        08C3  2248
                        08C3  2250          PUSHQ   R6                             ; Save registers
    50   03     90      08C6  2251          MOVB    #JNL_QIO,R0                    ; Set journal type = QIO
      1C01     30      08C9  2252          BSBW    XD_FILL_JNX                    ; Fill the journal buffer
    04 50      E9      08CC  2253          BLBC    R0,1$                          ; Br if error
F8 A6   6C     D0      08CF  2254          MOVL    P1(AP),-8(R6)                  ; Store the P1 parameter
                        08D3  2255 1$:      POPQ    R6                             ; Restore registers
                        08D6  2257
57   20 A3     B0      08D6  2258          MOVW    IRP$W_FUNC(R3),R7              ; Get entire function code
03 57   07     E1      08DA  2259          BBC     #IO$V_RD_MODEM,R7,3$           ; Br if not read modem request
      013F     31      08DE  2260          BRW     SENSE_MODEM                    ; Else, process read modem
                        08E1  2261
03 57   09     E1      08E1  2262 3$:      BBC     #IO$V_CTRL,R7,5$               ; Br if not controller request
      00EF     31      08E5  2263          BRW     SENSEMODE_CTRL                 ; Else, process controller req.
                        08E8  2264 ;
                        08E8  2265 ; Tributary sensemode request
                        08E8  2266 ;
      1603     30      08E8  2267 5$:      BSBW    XLATE                          ; Get CDB address
    64 50      E9      08EB  2268          BLBC    R0,30$                         ; Br if error
```

```
                31 57   08   E1  08EE  2269          BBC     #IO$V_RD_COUNT,R7,20$           ; Br if not read counters
                                 08F2  2270 ;
                                 08F2  2271 ; Read tributary counters - modifier RD_COUNT
                                 08F2  2272 ;
                    01DB   30    08F2  2273          BSBW    CHECK_P2                        ; Check P2 buffer
                50    14   3C    08F5  2274          MOVZWL  #SS$_BADPARAM,R0                ; Assume zero length buffer
             3A A3    51   B0    08F8  2275          MOVW    R1,IRP$L_MEDIA+2(R3)            ; Save user size of P2 buffer
                      54   13    08FC  2276          BEQL    30$                             ; Br if none
                51    3B   3C    08FE  2277          MOVZWL  S^#TRIB_CNT_BUFSIZ,R1           ; Get size of tributary P2 buf
                    0209   30    0901  2278          BSBW    ALLOC_P2BUF                     ; Allocate a buffer
                  4B 50    E9    0904  2279          BLBC    R0,30$                          ; Br if error
                51    2C A3    D0  0907  2280          MOVL    IRP$L_SVAPTE(R3),R1             ; Get system P2 buffer addr
             04 A1    52   D0    090B  2281          MOVL    R2,P2B_L_BUFFER(R1)             ; Save user's P2 buffer addr
          38 A3    FE   8F    90  090F  2282          MOVB    #254,IRP$L_MEDIA(R3)            ; Get 7 counters
             21 A3    07   90    0914  2283          MOVB    S^#XD_FC_V_RDTERR,IRP$B_XDFUNC(R3) ; Assume Read counts
             04 57    0A   E1    0918  2284          BBC     #IO$V_CLR_COUNT,R7,10$          ; Br if not clear counts
             21 A3    09   90    091C  2285          MOVB    S^#XD_FC_V_RCTERR,IRP$B_XDFUNC(R3) ; Else, Read and Clear
                    019B   31    0920  2286 10$:     BRW     QUEPKT                          ; Queue packet to driver
                                 0923  2287
             2E 57    06   E1    0923  2288 20$:     BBC     #IO$V_RD_MEM,R7,40$             ; Br if read tss request
                                 0927  2289 ;
                                 0927  2290 ; Read TSS address - modifier RD_MEM
                                 0927  2291 ;
                51    04 AC    9A  0927  2292          MOVZBL  P2(AP),R1                       ; Get TSS address
                   1F    51   91    092B  2293          CMPB    R1,S^#31                        ; Is TSS address too big?
                         22   1A    092E  2294          BGTRU   30$                             ; Br if yes - error
             3E A3    51   90    0930  2295          MOVB    R1,IRP$L_MEDIA+6(R3)            ; Save TSS address in IRP
                      51   04    9A  0934  2296          MOVZBL  S^#4,R1                         ; Check access to P1 buffer
                    01C0   30    0937  2297          BSBW    CHECK_P1                        ; Check P1 buffer
                    01D0   30    093A  2298          BSBW    ALLOC_P2BUF                     ; Allocate a P2 buffer
                  12 50    E9    093D  2299          BLBC    R0,30$                          ; Br if allocation error
                51    2C A3    D0  0940  2300          MOVL    IRP$L_SVAPTE(R3),R1             ; Get system buffer address
          04 A1    38 A3    D0  0944  2301          MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1) ; Save user P1 buffer VA
             38 A3    D4   0949  2302          CLRL    IRP$L_MEDIA(R3)                 ; Indicate NOT a counter IRP
          21 A3    0B   90    094C  2303          MOVB    S^#XD_FC_V_RDTSS,IRP$B_XDFUNC(R3) ; Set function request
                      CE   11    0950  2304          BRB     10$                             ; Queue packet
                                 0952  2305
                    F980   31    0952  2306 30$:     BRW     ABORTIO                         ; Abort the I/O request
                                 0955  2307
                                 0955  2308 ;
                                 0955  2309 ; Read tributary parameters - no modifier
                                 0955  2310 ;
                51    08   3C    0955  2311 40$:     MOVZWL  S^#8,R1                         ; Size of P1 buffer if present
                    0173   30    0958  2312          BSBW    CHECK_BUFS                      ; Check P1 and P2 buffers
             3C A3    51   B0    095B  2313          MOVW    R1,IRP$L_MEDIA+4(R3)            ; Save user P2 buffer length
                      5B   13    095F  2314          BEQL    60$                             ; Br if no P2 buffer present
                                 0961  2315
          51    006C 8F    3C    0961  2316          MOVZWL  #TRIB_PRM_BUFSIZ+6,R1           ; Set size of required buffer
                    0194   30    0966  2317          BSBW    ALLOC_P2BUF                     ; Allocate a P2 buffer
                  E6 50    E9    0969  2318          BLBC    R0,30$                          ; Br if error
                51    2C A3    D0  096C  2319          MOVL    IRP$L_SVAPTE(R3),R1             ; Get system P2 buffer address
             04 A1    52   D0    0970  2320          MOVL    R2,P2B_L_BUFFER(R1)             ; Set user's P2 buffer address
          51    F752 CF    9E  0974  2321          MOVAB   TRIB_PARAM,R1                   ; Get address of return table
                54    59   D0    0979  2322          MOVL    R9,R4                           ; Get CDB address
                    186A   30    097C  2323          BSBW    RETURN_P2                       ; Return the P2 parameters
             42 A3    01   B0    097F  2324          MOVW    S^#SS$_NORMAL,IRP$W_CSTATUS(R3) ; Assume success
                                 0983  2325 ;
```

```
                        0983  2326 ;       Return polling substate - only if device is inited.
                        0983  2327 ;
            00    E1    0983  2328          BBC     #UCB$V_XD_INITED,-                  ; Br if not inited
      10 68 A5          0985  2329                  UCB$W_DEVSTS(R5),45$                ;
            06    E1    0988  2330          BBC     #XM$V_CHR_CTRL,-                    ; Br if NOT a control station
      0B 44 A5          098A  2331                  UCB$L_DEVDEPEND(R5),45$             ;
   3E A3    02    90    098D  2332          MOVB    #2,IRP$L_MEDIA+6(R3)                ; Set TSS address (as read tss)
   21 A3    15    90    0991  2333          MOVB    #XD_FC_V_POLSS,IRP$B_XDFUNC(R3)     ; Set function request
         0126    31    0995  2334          BRW     QUEPKT                              ; Else, queue the packet
                        0998  2335
   32 A3    06    A2    0998  2336 45$:     SUBW    #6,IRP$W_BCNT(R3)                   ; No polling substate return
   50    32 A3    B0    099C  2337          MOVW    IRP$W_BCNT(R3),R0                   ; Assume user has good size buffer
   50    3C A3    B1    09A0  2338          CMPW    IRP$L_MEDIA+4(R3),R0                ; Is user buffer large enough?
            0E    1E    09A4  2339          BGEQU   50$                                 ; Br if yes
42 A3    0601    8F    B0    09A6  2340          MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3) ; Return partial success
   50    3C A3    B0    09AC  2341          MOVW    IRP$L_MEDIA+4(R3),R0                ; Set size of return
   32 A3    50    B0    09B0  2342          MOVW    R0,IRP$W_BCNT(R3)                   ;  ...
   50    50    10    78    09B4  2343 50$:     ASHL    #16,R0,R0                           ; Shift size of buffer return
   50    42 A3    B0    09B8  2344          MOVW    IRP$W_CSTATUS(R3),R0                ; Set success status
                        09BC  2345
   52    38 A3    D0    09BC  2346 60$:     MOVL    IRP$L_MEDIA(R3),R2                  ; Retrieve P1 buffer address
            08    13    09C0  2347          BEQL    70$                                 ; Br if none
   62    40 A5    7D    09C2  2348          MOVQ    UCB$B_DEVCLASS(R5),(R2)             ; Else, return characteristics
   04 A2    69    C8    09C6  2349          BISL    CDB_L_DEVDEPEND(R9),4(R2)           ;  ...
      51    69    D0    09CA  2350 70$:     MOVL    CDB_L_DEVDEPEND(R9),R1              ; Get device dependend info
   51    44 A5    C8    09CD  2351          BISL    UCB$L_DEVDEPEND(R5),R1              ; ..from UCB also
00000000'GF    17    09D1  2352          JMP     G^EXE$FINISHIO                      ; Complete the I/O request
```

```
                                                09D7  2354          .SBTTL  SENSEMODE_CTRL,  Perform SENSEMODE FDT processing for controller
                                                09D7  2355
                                                09D7  2356 ;++
                                                09D7  2357 ; SENSEMODE_CTRL - Perform SENSEMODE FDT processing for controller
                                                09D7  2358 ;
                                                09D7  2359 ; Functional description:
                                                09D7  2360 ;
                                                09D7  2361 ; This routine performs all FDT checking for a controller SENSEMODE request.
                                                09D7  2362 ; The P2 buffer if present is check for write access and if okay, a system
                                                09D7  2363 ; buffer is allocated for temporarily saving the needed information. The
                                                09D7  2364 ; P1 sensemode information is returned through IRP$L_MEDIA and IRP$L_MEDIA+4.
                                                09D7  2365 ;
                                                09D7  2366 ; Inputs:
                                                09D7  2367 ;
                                                09D7  2368 ;       R3 = IRP address
                                                09D7  2369 ;       R4 = PCB address
                                                09D7  2370 ;       R5 = UCB address
                                                09D7  2371 ;       R7 = Function code
                                                09D7  2372 ;
                                                09D7  2373 ; Outputs:
                                                09D7  2374 ;
                                                09D7  2375 ;       R0 = status return for request
                                                09D7  2376 ;
                                                09D7  2377 ;       R3-R5 are preserved.
                                                09D7  2378 ;
                                                09D7  2379 ;--
                                                09D7  2380
                                                09D7  2381 SENSEMODE_CTRL:                                      ; Process controller sensemode FDT
                31 57    08    E1               09D7  2382          BBC     #IO$V_RD_COUNT,R7,20$              ; Br if not read counters
                                                09D8  2383 ;
                                                09D8  2384 ; Read controller counters - modifier RD_COUNT
                                                09D8  2385 ;
                     00F2    30                 09DB  2386          BSBW    CHECK_P2                           ; Check P2 buffer
                   50    14    3C               09DE  2387          MOVZWL  #SS$_BADPARAM,R0                   ; Assume zero length buffer
             3A A3       51    B0               09E1  2388          MOVW    R1,IRP$L_MEDIA+2(R3)               ; Save user P2 buffer size
                           54    13               09E5  2389          BEQL    30$                                ; Br if no P2 buffer
                   51      0A    3C               09E7  2390          MOVZWL  S^#LINE_CNT_BUFSIZ,R1              ; Get size of controller counts
                        0120     30               09EA  2391          BSBW    ALLOC_P2BUF                        ; Allocate a buffer
                     4B 50       E9               09ED  2392          BLBC    R0,30$                             ; Br if error
                   51    2C A3    D0               09F0  2393          MOVL    IRP$L_SVAPTE(R3),R1                ; Get system P2 buffer address
                04 A1       52    D0               09F4  2394          MOVL    R2,P2B_L_BUFFER(R1)                ; Save user P2 buffer address
          38 A3       60    8F    90               09F8  2395          MOVB    #<3@5>,IRP$L_MEDIA(R3)             ; Get 2 counters
             21 A3       08    90               09FD  2396          MOVB    S^#XD_FC_V_RDGERR,IRP$B_XDFUNC(R3) ; Assume only read counts
                04 57    0A    E1               0A01  2397          BBC     #IO$V_CLR_COUNT,R7,10$             ; Br if not clear count request
             21 A3       0A    90               0A05  2398          MOVB    S^#XD_FC_V_RCGERR,IRP$B_XDFUNC(R3) ; Else, Read and Clear
                     00B2    31               0A09  2399 10$:      BRW     QUEPKT                             ; Queue packet to driver
                                                0A0C  2400
                2E 57    06    E1               0A0C  2401 20$:      BBC     #IO$V_RD_MEM,R7,40$                ; Br if read tss request
                                                0A10  2402 ;
                                                0A10  2403 ; Read GSS address - modifier RD_MEM
                                                0A10  2404 ;
                51    04 AC    9A               0A10  2405          MOVZBL  P2(AP),R1                          ; Get GSS address
                   1F    51    91               0A14  2406          CMPB    R1,S^#31                           ; Is GSS address too big?
                         22    1A               0A17  2407          BGTRU   30$                                ; Br if yes - error
             3E A3       51    90               0A19  2408          MOVB    R1,IRP$L_MEDIA+6(R3)               ; Save GSS address in IRP
                   51    04    9A               0A1D  2409          MOVZBL  S^#4,R1                            ; Check access to P1 buffer
                        00D7     30               0A20  2410          BSBW    CHECK_P1                           ; Check P1 buffer
```

```
                00E7   30  0A23  2411          BSBW    ALLOC_P2BUF                         ; Allocate a P2 buffer
              12 50    E9  0A26  2412          BLBC    R0,30$                              ; Br if error
      51      2C A3    D0  0A29  2413          MOVL    IRP$L_SVAPTE(R3),R1                 ; Get system buffer address
   04 A1      38 A3    D0  0A2D  2414          MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1)    ; Set user buffer VA
      38 A3   0C       90  0A32  2415          CLRL    IRP$L_MEDIA(R3)                     ; Indicate NOT a counter IRP
   21 A3      0C       90  0A35  2416          MOVB    S^#XD_FC_V_RDGSS,IRP$B_XDFUNC(R3)   ; Set function request
              CE       11  0A39  2417          BRB     10$                                 ; Queue packet
                           0A3B  2418
              F897     31  0A3B  2419 30$:     BRW     ABORTIO                             ; Abort the I/O request
                           0A3E  2420
                           0A3E  2421 ;
                           0A3E  2422 ; Read controller parameters - no modifier
                           0A3E  2423 ;
      51      08       3C  0A3E  2424 40$:     MOVZWL  S^#8,R1                             ; Size of P1 buffer if present
                00BA   30  0A41  2425          BSBW    CHECK_BUFS                          ; Check P1 and P2 buffers
   3C A3      51       D0  0A44  2426          MOVL    R1,IRP$L_MEDIA+4(R3)                ; Save user P2 buffer length
                42       13  0A48  2427          BEQL    60$                                 ; Br if no P2 buffer
   51      0042 8F     3C  0A4A  2428          MOVZWL  #LINE_PRM_BUFSIZ,R1                 ; Set size of required buffer
                00BB   30  0A4F  2429          BSBW    ALLOC_P2BUF                         ; Allocate a P2 buffer
                E6 50    E9  0A52  2430          BLBC    R0,30$                              ; Br if error
      51      2C A3    D0  0A55  2431          MOVL    IRP$L_SVAPTE(R3),R1                 ; Get system P2 buffer address
   04 A1      52       D0  0A59  2432          MOVL    R2,P2B_L_BUFFER(R1)                 ; Set user's P2 buffer address
   51      F617 CF     9E  0A5D  2433          MOVAB   LINE_PARAM,R1                       ; Get address of return table
              54   55   D0  0A62  2434          MOVL    R5,R4                               ; Get UCB address
                1781   30  0A65  2435          BSBW    RETURN_P2                           ; Return the P2 buffer data
   42 A3      01       B0  0A68  2436          MOVW    S^#SS$_NORMAL,IRP$W_CSTATUS(R3)     ; Assume success
      50   32 A3       B0  0A6C  2437          MOVW    IRP$W_BCNT(R3),R0                   ; Assume user has good size buffer
      50   3C A3       B1  0A70  2438          CMPW    IRP$L_MEDIA+4(R3),R0                ; Is user buffer large enough?
                0E     1E  0A74  2439          BGEQU   50$                                 ; Br if yes
42 A3      0601 8F     B0  0A76  2440          MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3)    ; Return partial success
      50   3C A3       B0  0A7C  2441          MOVW    IRP$L_MEDIA+4(R3),R0                ; Set size of return
   32 A3      50       B0  0A80  2442          MOVW    R0,IRP$W_BCNT(R3)                   ;                ...
   50      50   10     78  0A84  2443 50$:     ASHL    #16,R0,R0                           ; Shift return buffer size
      50   42 A3       B0  0A88  2444          MOVW    IRP$W_CSTATUS(R3),R0                ; Set return status
                           0A8C  2445
      52   38 A3       D0  0A8C  2446 60$:     MOVL    IRP$L_MEDIA(R3),R2                  ; Retrieve P1 buffer address
                04     13  0A90  2447          BEQL    70$                                 ; Br if none
      62   40 A5       7D  0A92  2448          MOVQ    UCB$B_DEVCLASS(R5),(R2)             ; Else, return characteristics
      51   44 A5       D0  0A96  2449 70$:     MOVL    UCB$L_DEVDEPEND(R5),R1              ; Get device dependend info
00000000'GF       17   0A9A  2450          JMP     G^EXE$FINISHIO                      ; Complete the I/O request
```

```
                     0AA0  2452          .SBTTL  SENSE_MODEM,  Perform SENSEMODE READ_MODEM FDT processing
                     0AA0  2453
                     0AA0  2454 ;++
                     0AA0  2455 ; SENSE_MODEM - Perform SENSEMODE READ_MODEM FDT processing
                     0AA0  2456 ;
                     0AA0  2457 ; Functional description:
                     0AA0  2458 ;
                     0AA0  2459 ; This routine performs all FDT checking for a SENSEMODE read modem request.
                     0AA0  2460 ; The P1 buffer is checked for write access and the request is queued to the
                     0AA0  2461 ; driver. When complete the P1 sensemode information is returned in IRP$L_MEDIA
                     0AA0  2462 ; and IRP$L_MEDIA+4.
                     0AA0  2463 ;
                     0AA0  2464 ; Inputs:
                     0AA0  2465 ;
                     0AA0  2466 ;       R3 = IRP address
                     0AA0  2467 ;       R4 = PCB address
                     0AA0  2468 ;       R5 = UCB address
                     0AA0  2469 ;       R7 = Function code
                     0AA0  2470 ;
                     0AA0  2471 ; Outputs:
                     0AA0  2472 ;
                     0AA0  2473 ;       R0 = status return for request
                     0AA0  2474 ;
                     0AA0  2475 ;       R3-R5 are preserved.
                     0AA0  2476 ;
                     0AA0  2477 ;--
                     0AA0  2478
                     0AA0  2479 SENSE_MODEM:                                    ; Process read modem request
51   04   3C           0AA0  2480          MOVZWL  S^#4,R1                     ; Size of P1 buffer
     55   10           0AA3  2481          BSBB    CHECK_P1                    ; Check access to P1
                       0AA5  2482                                              ; (no return on no access)
          0065   30    0AA5  2483          BSBW    ALLOC_P2BUF                 ; Allocate a P2 buffer
        03 50    E8    0AA8  2484          BLBS    R0,10$                      ; Br if success
          F827   31    0AAB  2485          BRW     ABORTIO                     ; Else, abort the I/O request
                       0AAE  2486
   51    2C A3   D0    0AAE  2487 10$:     MOVL    IRP$L_SVAPTE(R3),R1         ; Get system buffer address
04 A1    38 A3   D0    0AB2  2488          MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1) ; Set user buffer VA
         38 A3   D4    0AB7  2489          CLRL    IRP$L_MEDIA(R3)             ; Indicate NOT a counter IRP
   21 A3    06   90    0ABA  2490          MOVB    S^#XD_FC_V_RDMODEM,IRP$B_XDFUNC(R3) ; Set function request
                       0ABE  2491          ;BRB    QUEPKT                      ; Give request to driver
                       0ABE  2492
                       0ABE  2493 ;
                       0ABE  2494 ; I/O Request packet to driver
                       0ABE  2495 ;
                       0ABE  2496 QUEPKT:                                      ; Queue packet to driver
                       0ABE  2497          SETIPL  UCB$B_FIPL(R5)              ; Raise IPL to fork IPL
00000000'GF    16     0AC2  2498          JSB     G^IOC$INITIATE              ; Initiate I/O request
00000000'GF    17     0AC8  2499          JMP     G^EXE$QIORETURN             ; Lower IPL, and RET
```

```
                        0ACE  2501          .SBTTL  CHECK_BUFS,  Check P1 and P2 buffers for write access
                        0ACE  2502
                        0ACE  2503 ;++
                        0ACE  2504 ; CHECK_BUFS - Check P1 and P2 buffers for write access
                        0ACE  2505 ;
                        0ACE  2506 ; Functional description:
                        0ACE  2507 ;
                        0ACE  2508 ; This routines checks the P1 and P2 buffers for write access if supplied.
                        0ACE  2509 ;
                        0ACE  2510 ; Inputs:
                        0ACE  2511 ;
                        0ACE  2512 ;       R1 = Size of P1 buffer needed for write access
                        0ACE  2513 ;       R3 = IRP address
                        0ACE  2514 ;       R4 = PCB address
                        0ACE  2515 ;       R5 = UCB address
                        0ACE  2516 ;       R7 = Function code
                        0ACE  2517 ;       R9 = CDB address
                        0ACE  2518 ;
                        0ACE  2519 ; Outputs:
                        0ACE  2520 ;
                        0ACE  2521 ;       R1 = Length of P2 buffer (zero if no P2 buffer)
                        0ACE  2522 ;       R2 = Address of P2 buffer in user's process space
                        0ACE  2523 ;
                        0ACE  2524 ;       R0-R2 are destroyed.
                        0ACE  2525 ;
                        0ACE  2526 ;       No RETURN on NO ACCESS
                        0ACE  2527 ;
                        0ACE  2528 ; Implicit Outputs:
                        0ACE  2529 ;
                        0ACE  2530 ;       IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                        0ACE  2531 ;
                        0ACE  2532 ;--
                        0ACE  2533
                        0ACE  2534 CHECK_BUFS:
          2A    10      0ACE  2535          BSBB    CHECK_P1                        ; Check P1 buffer
                        0AD0  2536 CHECK_P2:
          51    D4      0AD0  2537          CLRL    R1                              ; Assume no P2 buffer desc
    52    04 AC   D0      0AD2  2538          MOVL    P2(AP),R2                       ; Get address of P2 desc
          1B    13      0AD6  2539          BEQL    10$                             ; Br if no P2
                        0AD8  2540          IFNORD  #8,(R2),ACCESS                  ; Br if no access
    51    62    3C      0ADE  2541          MOVZWL  (R2),R1                         ; Get length of buffer
    51    01    CA      0AE1  2542          BICL    #1,R1                           ; Must be multiple of 2 bytes
          0D    13      0AE4  2543          BEQL    10$                             ; Br if zero
    50    04 A2   D0      0AE6  2544          MOVL    DSC$A_POINTER(R2),R0            ; Get buffer address
00000000'GF   16      0AEA  2545          JSB     G^EXE$READCHK                   ; Check write access to buffer
                        0AF0  2546                                                  ; (no return no access)
                        0AF0  2547                                                  ; Also sets IRP$V_FUNC in IRP
    52    50    D0      0AF0  2548          MOVL    R0,R2                           ; Copy buffer address
                05      0AF3  2549 10$:     RSB                                     ; Return to caller
                        0AF4  2550
    50    0C    3C      0AF4  2551 ACCESS:  MOVZWL  #SS$_ACCVIO,R0                  ; Return access violation
        F7DB    31      0AF7  2552          BRW     ABORTIO                         ; Abort the I/O request
```

```
                                .SBTTL  CHECK_P1,  Check P1 buffer address for write access

                        ;++
                        ; CHECK_P1 - Check P1 buffer address for write access
                        ;
                        ; Functional description:
                        ;
                        ;  This routine checks the P1 buffer and if okay, the buffer address
                        ;  is saved in IRP$L_MEDIA of the IRP.
                        ;
                        ; Inputs:
                        ;
                        ;       R1 = Size of buffer for write access
                        ;       R3 = IRP address
                        ;       R4 = PCB address
                        ;       R5 = UCB address
                        ;       R7 = Function code
                        ;       R9 = CDB address
                        ;
                        ; Outputs:
                        ;
                        ;       R0 is destroyed.
                        ;
                        ;       No RETURN on NO ACCESS.
                        ;
                        ; Implicit Outputs:
                        ;
                        ;       IRP$L_MEDIA(R3) = User P1 buffer address.
                        ;       IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                        ;
                        ;--

                        CHECK_P1:
38 A3   D4              CLRL    IRP$L_MEDIA(R3)         ; Assume no P1 buffer
50 6C   D0              MOVL    P1(AP),R0               ; Get address of user buffer
   0A   13              BEQL    10$                     ; Br if none
00000000'GF 16          JSB     G^EXE$READCHK           ; Check access to buffer
                                                        ; (No return - no access)
38 A3   50      D0      MOVL    R0,IRP$L_MEDIA(R3)      ; Save P1 buffer address in IRP
                05      RSB                             ; Return to caller
```

```
                     0B0D  2595          .SBTTL  ALLOC_P2BUF,  Allocate a P2 buffer and charge user's quota
                     0B0D  2596
                     0B0D  2597 ;++
                     0B0D  2598 ; ALLOC_P2BUF - Allocate a P2 buffer and charge user's quota
                     0B0D  2599 ;
                     0B0D  2600 ; Functional description:
                     0B0D  2601 ;
                     0B0D  2602 ; This routine allocates a system buffer and returns the address in the IRP at
                     0B0D  2603 ; IRP$L_SVAPTE.  The size of the allocation, including buffer header must
                     0B0D  2604 ; be at least 24 bytes in length.
                     0B0D  2605 ;
                     0B0D  2606 ; Inputs:
                     0B0D  2607 ;
                     0B0D  2608 ;       R1 = Size of allocation desired
                     0B0D  2609 ;       R3 = IRP address
                     0B0D  2610 ;
                     0B0D  2611 ; Outputs:
                     0B0D  2612 ;
                     0B0D  2613 ;       R0 = status of request
                     0B0D  2614 ;
                     0B0D  2615 ;       R1-R5 are preserved.
                     0B0D  2616 ;
                     0B0D  2617 ; Implicit Outputs:
                     0B0D  2618 ;
                     0B0D  2619 ;       IRP$L_SVAPTE(R3) = address of system buffer
                     0B0D  2620 ;       IRP$W_BOFF(R3) = byte count charged to user's process
                     0B0D  2621 ;       IRP$W_BCNT(R3) = original byte count requested
                     0B0D  2622 ;
                     0B0D  2623 ;       All parts of the P2 buffer header are initialized, except for the
                     0B0D  2624 ;       user's P2 buffer address.
                     0B0D  2625 ;
                     0B0D  2626 ;--
                     0B0D  2627
                     0B0D  2628 ALLOC_P2BUF:                                 ; Allocate a non-paged buffer
          51    D5   0B0D  2629          TSTL    R1                         ; Zero length buffer?
          50    13   0B0F  2630          BEQL    30$                        ; Br if yes
          0E    BB   0B11  2631          PUSHR   #^M<R1,R2,R3>              ; Save registers
    32 A3     51    B0   0B13  2632          MOVW    R1,IRP$W_BCNT(R3)          ; Save original byte count
       0C     51    D1   0B17  2633          CMPL    R1,S^#24-P2B_C_LENGTH      ; Is buffer big enough?
          03    1A   0B1A  2634          BGTRU   5$                         ; Br if yes
       51    0C    D0   0B1C  2635          MOVL    S^#24-P2B_C_LENGTH,R1      ; Else, set size to minimum
       51    0C    C0   0B1F  2636 5$:      ADDL2   S^#P2B_C_LENGTH,R1         ; Add in size of header
00000000'GF     16   0B22  2637          JSB     G^EXE$BUFQUOPRC            ; Check for buffered quota
          0D 50    E9   0B28  2638          BLBC    R0,10$                     ; Branch if quota bad
                        0B2B  2639 ;
                        0B2B  2640 ; Quota OKAY, allocate buffer and copy info.
                        0B2B  2641 ;
          51    DD   0B2B  2642          PUSHL   R1                         ; Save size to charge user
00000000'GF     16   0B2D  2643          JSB     G^EXE$ALONONPAGED          ; Go allocate a buffer
          05 50    E8   0B33  2644          BLBS    R0,20$                     ; Br if success
          8E    D5   0B36  2645          TSTL    (SP)+                      ; Pop saved size
          0E    BA   0B38  2646 10$:     POPR    #^M<R1,R2,R3>              ; Restore registers
                05    0B3A  2647          RSB                                ; Return with error code in R0
                        0B3B  2648 ;
                        0B3B  2649 ; System buffer allocated decrement user's quota
                        0B3B  2650 ;
          53 8ED0   0B3B  2651 20$:     POPL    R3                         ; Restore user quota charge
```

```
                62    0C  A2    9E   0B3E  2652          MOVAB   P2B_T_DATA(R2),P2B_L_POINTER(R2) ; Set address to start of data
                08  A2    53    B0   0B42  2653          MOVW    R3,P2B_W_SIZE(R2)                ; Save buffer size in buffer
                0A  A2    13    90   0B46  2654          MOVB    S^#DYN$C_BUFIO,P2B_B_TYPE(R2)    ; Set structure type
                    50    52    D0   0B4A  2655          MOVL    R2,R0                            ; Save P2 char buf addr
            52      0080  C4    D0   0B4D  2656          MOVL    PCB$L_JIB(R4),R2                 ; Get JIB address
                20  A2    53    C2   0B52  2657          SUBL    R3,JIB$L_BYTCNT(R2)              ; Decrement user's quota
                          0E    BA   0B56  2658          POPR    #^M<R1,R2,R3>                    ; Restore registers
                2C  A3    50    D0   0B58  2659          MOVL    R0,IRP$L_SVAPTE(R3)              ; Save P2 buffer address in IRP
            30  A3      08  A0    B0   0B5C  2660          MOVW    P2B_W_SIZE(R0),IRP$W_BOFF(R3)    ; Return buffer size in IRP
                    50    01    3C   0B61  2661 30$:      MOVZWL  S^#SS$_NORMAL,R0                 ; Set success
                                  05   0B64  2662          RSB                                      ; Return to caller
```

```
                        0B65  2664          .SBTTL  STARTIO,  Start I/O routine
                        0B65  2665
                        0B65  2666 ;++
                        0B65  2667 ; STARTIO - Start a transmit, receive, or set mode operation
                        0B65  2668 ;
                        0B65  2669 ; Functional description:
                        0B65  2670 ;
                        0B65  2671 ; This routine is called when an IRP is given to EXE$QIODRVPKT.
                        0B65  2672 ; From here, depending on the type of function., the appropriate
                        0B65  2673 ; routine is called.
                        0B65  2674 ;
                        0B65  2675 ; Inputs:
                        0B65  2676 ;
                        0B65  2677 ;       R3 = IRP address (I/O request packet)
                        0B65  2678 ;       R5 = UCB address (unit control block)
                        0B65  2679 ;       R9 = CDB address (only if circuit request)
                        0B65  2680 ;
                        0B65  2681 ;
                        0B65  2682 ;       IPL = FIPL
                        0B65  2683 ;
                        0B65  2684 ; Outputs:
                        0B65  2685 ;
                        0B65  2686 ;       R3,R5 are preserved.
                        0B65  2687 ;
                        0B65  2688 ;       The routine must preserve all registers except R0-R2 and R4.
                        0B65  2689 ;
                        0B65  2690 ;--
                        0B65  2691
                        0B65  2692 STARTIO:                                     ; Process an I/O packet
                        0B65  2694          PUSHQ   R6                          ; Save registers
     50    04    90     0B68  2695          MOVB    #JNL_QUE,R0                 ; Set journal type = QUE
        195F    30     0B6B  2696          BSBW    XD_FILL_JNX                 ; Fill the journal buffer
                        0B6E  2697          POPQ    R6                          ; Restore registers
                        0B71  2699
     51    21 A3    9A  0B71  2700          MOVZBL  IRP$B_XDFUNC(R3),R1         ; Get the function code
        01    51    91  0B75  2701          CMPB    R1,S^#XD_FC_V_INITC         ; Startup device?
              0A    13  0B78  2702          BEQL    10$                         ; Branch if so
     50    0084 8F    3C  0B7A  2703          MOVZWL  #SS$_DEVOFFLINE,R0          ; Assume device not running
              10    E0  0B7F  2704          BBS     #XM$V_ERR_FATAL,-           ; Br if hung
           42 44 A5       0B81  2705                  UCB$L_DEVDEPEND(R5),DRV_DONE ;
                        0B84  2706 10$:     $DISPATCH       R1,TYPE=B,-         ; Dispatch on function request
                        0B84  2707          <-      ;function       action
                        0B84  2708
                        0B84  2709                  <XD_FC_V_INITT  START_TRIB>,- ; Start tributary
                        0B84  2710                  <XD_FC_V_INITC  START_DEV>,- ; Start DMP11 controller
                        0B84  2711                  <XD_FC_V_STOPT  TRIB_ACTIVE>,- ; Stop tributary
                        0B84  2712                  <XD_FC_V_STOPC  DEV_ACTIVE>,- ; Stop DMP11 controller
                        0B84  2713                  <XD_FC_V_CHGC   DEV_ACTIVE>,- ; Change controller parameters
                        0B84  2714                  <XD_FC_V_CHGT   TRIB_ACTIVE>,- ; Change tributary parameters
                        0B84  2715                  <XD_FC_V_RDMODEM DEV_ACTIVE>,- ; Read modem register
                        0B84  2716                  <XD_FC_V_RDTERR TRIB_ERRS>,- ; Read tributary error counts
                        0B84  2717                  <XD_FC_V_RDGERR DEV_ACTIVE>,- ; Read global error counts
                        0B84  2718                  <XD_FC_V_RCTERR TRIB_ERRS>,- ; Read and clear tributary ctrs
                        0B84  2719                  <XD_FC_V_RCGERR DEV_ACTIVE>,- ; Read and clear global counts
                        0B84  2720                  <XD_FC_V_RDTSS  TRIB_ACTIVE>,- ; Read TSS location
                        0B84  2721                  <XD_FC_V_RDGSS  DEV_ACTIVE>,- ; Read GSS location
                        0B84  2722                  <XD_FC_V_WTMODEM DEV_ACTIVE>,- ; Write modem request
```

```
                     0B84  2723                  <XD_FC_V_POLSS  TRIB_ACTIVE>,-  ; Read polling sub-state
                     0B84  2724          >
                     0B84  2725 ;
                     0B84  2726 ; Other request types - if we fall thru case instruction.
                     0B84  2727 ;
                     0B84  2728          BUG_CHECK NOBUFPCKT,FATAL               ; Fatal error
                     0B88  2729 ;
                     0B88  2730 ; Start up controller
                     0B88  2731 ;
                     0B88  2732 START_DEV:                                       ; Startup controller
      0F     10      0B88  2733          BSBB    STARTUP                         ; Startup device
01 50        E9      0B8A  2734          BLBC    R0,10$                          ; Br if error
               05    0B8D  2735          RSB                                     ; Return to caller
                     0B8E  2736 ;
                     0B8E  2737 ; Error on startup - shutdown the controller
                     0B8E  2738 ;
      50     DD      0B8E  2739 10$:     PUSHL   R0                              ; Else, save status return
    1179     30      0B90  2740          BSBW    SHUTDOWN_DEV                    ; Shutdown the device
      50 8ED0        0B93  2741          POPL    R0                              ; Restore status return
                     0B96  2742          ;BRB    DRV_DONE                        ; Complete request in error
                     0B96  2743
                     0B96  2744 DRV_DONE.                                        ; Driver request complete
    08F8     31      0B96  2745          BRW     REQ_COM_ERR                     ; Complete the request
```

```
                                0BC9  2747          .SBTTL  STARTUP,  Device initialization routine
                                0BC9  2748
                                0BC9  2749 ;++
                                0BC9  2750 ; STARTUP - Device initialization routine
                                0BC9  2751 ;
                                0BC9  2752 ; Functional description:
                                0BC9  2753 ;
                                0BC9  2754 ; The configuration is read from the P1 and P2 buffers and a MODE DEFINITION
                                0BC9  2755 ; command is issued to the DMP11 device.  The buffers which have been set
                                0BC9  2756 ; up for the common receive pool are then given to the device.
                                0BC9  2757 ;
                                0BC9  2758 ; Inputs:
                                0BC9  2759 ;
                                0BC9  2760 ;       R3 = IRP address
                                0BC9  2761 ;       R5 = UCB address
                                0BC9  2762 ;
                                0BC9  2763 ;       IPL = FIPL
                                0BC9  2764 ;
                                0BC9  2765 ; Outputs:
                                0BC9  2766 ;
                                0BC9  2767 ;       R0 = status return for startup operation
                                0BC9  2768 ;
                                0BC9  2769 ;       R3,R5 are preserved
                                0BC9  2770 ;
                                0BC9  2771 ;--
                                0BC9  2772
                                0BC9  2773 STARTUP:                                     ; Controller startup processing
                      00   E1   0BC9  2774          BBC     #UCB$V_XD_INITED,-        ; Br if not inited yet
                 0C 68 A5        0BCB  2775                  UCB$W_DEVSTS(R5),1$       ;
               50   02C4 8F   3C  0BCE  2776          MOVZWL  #SS$_DEVACTIVE,R0         ; Else, device already active
                      08EB   30  0BD3  2777          BSBW    REQ_COM_ERR               ; Complete request in error
                    50   01   9A  0BD6  2778          MOVZBL  S^#SS$_NORMAL,R0          ; DO NOT STOP CONTROLLER!
                                05  0BD9  2779          RSB                               ; Return to caller
                                0BDA  2780
F56F CF           42 A5   B1   0BDA  2781 1$:       CMPW    UCB$W_DEVBUFSIZ(R5),MAX_W_BUFFER; Is buffer too big?
                                04  1B  0BE0  2782          BLEQU   2$                        ; Br if not, okay
                    50   14   3C  0BE2  2783          MOVZWL  #SS$_BADPARAM,R0          ; Else, set error condition
                                05  0BE5  2784          RSB                               ; Return to caller
                                0BE6  2785
                                0BE6  2786 2$:        ;
                                0BE6  2787          ; For u-VAX I, we must have a contiguous buffer area!
                                0BE6  2788          ;
                                0BE6  2789          CPUDISP <<790,9$>,-
                                0BE6  2790                   <780,9$>,-
                                0BE6  2791                   <750,9$>,-
                                0BE6  2792                   <730,9$>,-
                                0BE6  2793                   <UV1,3$>>
                                0C00  2794 3$:
             52   012C C5   D0   0C00  2795          MOVL    UCB$L_XD_UV1BUF(R5),R2    ; Is the buffer area present?
                          06   12  0C05  2796          BNEQ    5$                        ; Br if not, error
             50   0124 8F   3C   0C07  2797          MOVZWL  #SS$_INSFMEM,R0           ; Else, set error condition
                                05  0C0C  2798          RSB                               ; Return to caller
                                0C0D  2799
                                0C0D  2800 5$:        ;
                                0C0D  2801          ; Compute physical/virtual address of buffers in buffer area.
                                0C0D  2802          ;
               52   0C A2   9E   0C0D  2803          MOVAB   12(R2),R2                 ; Get start of buffer area
```

```
                50   52    15    09    EF  0C11  2804          EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R0       ; Get virtual page number
                51    00000000'GF    D0  0C16  2805          MOVL    G^MMG$GL_SPTBASE,R1             ; Get the base address of SPTs
                        50    6140   D0  0C1D  2806          MOVL    (R1)[R0],R0                     ; Get the PTE contents
          51    52    FFFFFE00 8F    CB  0C21  2807          BICL3   #^C<VA$M_BYTE>,R2,R1            ; Get buffer offset (BA00-BA08)
                                     0C29  2808          ASSUME  PTE$S_PFN GE 13
                51    0D    09    50  F0  0C29  2809          INSV    R0,#9,#13,R1                    ; Copy BA09-BA21
                                    50   D4  0C2E  2810          CLRL    R0                              ; Use R0 as receive buffer index
                   00FC C540    51    D0  0C30  2811 7$:      MOVL    R1,UCB$L_XD_RCV_PA(R5)[R0]      ; Save RECV Physical address
                   0114 C540    52    D0  0C36  2812          MOVL    R2,UCB$L_XD_RCV_VA(R5)[R0]      ; Save RECV Virtual address
                51    00000400 8F    C0  0C3C  2813          ADDL    #MAX_BUFSIZ_UV1,R1              ; Skip to next buffer
                52    00000400 8F    C0  0C43  2814          ADDL    #MAX_BUFSIZ_UV1,R2              ;     ...
                      E2 50    04    F2  0C4A  2815          AOBLSS  #MAX_RCV_UVT,R0,7$              ; Loop if more
                                    50   D4  0C4E  2816          CLRL    R0                              ; Use R0 as xmit buffer index
                   010C C540    51    D0  0C50  2817 8$:      MOVL    R1,UCB$L_XD_XMT_PA(R5)[R0]      ; Save XMIT Physical address
                   0124 C540    52    D0  0C56  2818          MOVL    R2,UCB$L_XD_XMT_VA(R5)[R0]      ; Save XMIT Virtual address
                51    00000400 8F    C0  0C5C  2819          ADDL    #MAX_BUFSIZ_UV1,R1              ; Skip to next buffer
                52    00000400 8F    C0  0C63  2820          ADDL    #MAX_BUFSIZ_UV1,R2              ;     ...
                      E2 50    02    F2  0C6A  2821          AOBLSS  #MAX_XMT_UVT,R0,8$              ; Loop if more
                                     0C6E  2822
                                     0C6E  2823 9$:
          44 A5   18    08    00    F0  0C6E  2824          INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5)   ; Reset status and error flags
                02A0 C5    0C A3    D0  0C74  2825          MOVL    IRP$L_PID(R3),UCB$L_XD_PID(R5)  ; Save starter's PID
                                     0C7A  2826          ASSUME  UCB$B_XD_OUTTIM EQ UCB$B_XD_INTIM+1
                         02A5 C5    B4  0C7A  2827          CLRW    UCB$B_XD_INTIM(R5)              ; Init timer cells
                                     0C7E  2828          SETBIT  #UCB$V_XD_INITING,UCB$W_DEVSTS(R5) ; Indicate device initing
                                     0C83  2829 ;
                                     0C83  2830 ; Initialize the buffer and I/O request queue heads
                                     0C83  2831 ;
                      50    06    3C  0C83  2832          MOVZWL  #UCB$C_XD_QUEUES,R0             ; Set number of queue heads
          52    0094 C5    9E    0C86  2833          MOVAB   UCB$Q_XD_QUEUES(R5),R2          ; Set address of first head
                82    62    9E  0C8B  2834 10$:     MOVAB   (R2),(R2)+                      ; Init forward link
            82    FC A2    9E  0C8E  2835          MOVAB   -4(R2),(R2)+                    ; Init backward link
                  F6 50    F5  0C92  2836          SOBGTR  R0,10$                          ; Br if more queues
                                     0C95  2837 ;
                                     0C95  2838 ; Initialize the transmit and receive mapping info vector.
                                     0C95  2839 ;
                0298 C5    D4  0C95  2840          CLRL    UCB$L_XD_RUN(R5)                ; Initialize tribs in run
                029C C5    D4  0C99  2841          CLRL    UCB$L_XD_HALT(R5)               ; Initialize halted tribs
                0134 C5    94  0C9D  2842          CLRB    UCB$B_XD_TRB_CNT(R5)            ; Init number of active tribs
                50    0E    3C  0CA1  2843          MOVZWL  #MAX_RCV+MAX_XMT,R0             ; Set number of rcv and xmit slots
                                     0CA4  2844          ASSUME  UCB$L_XD_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XD_XMT_MAP
          51    00C4 C5    DE    0CA4  2845          MOVAL   UCB$L_XD_RCV_MAP(R5),R1         ; Get map vector address
                81    01    CE  0CA9  2846 15$:     MNEGL   #1,(R1)+                        ; Indicate no map info.
                  FA 50    F5  0CAC  2847          SOBGTR  R0,15$                          ; Br if more map slots
                                     0CAF  2848 ;
                                     0CAF  2849 ; Initialize vector areas of UCB
                                     0CAF  2850 ;
          51    0138 C5    DE    0CAF  2851          MOVAL   UCB$L_XD_VECS(R5),R1            ; Get address of start of vectors
          52    0058 8F    3C    0CB4  2852          MOVZWL  #UCB$C_XD_VECS,R2               ; Get number of longwords
                      81    D4  0CB9  2853 20$:     CLRL    (R1)+                           ; Initialize vector area
                  FB 52    F5  0CBB  2854          SOBGTR  R2,20$                          ; Loop if more
                                     0CBE  2855
                32 A3    B4  0CBE  2856          CLRW    IRP$W_BCNT(R3)                  ; No read buffer for I/O post
      0136 C5   42 A3    B0  0CC1  2857          MOVW    IRP$W_QUOTA(R3),UCB$W_XD_QUOTA(R5) ; Store receive buffer quota
                42 A3    B4  0CC7  2858          CLRW    IRP$W_QUOTA(R3)                 ; No buffer quota to return
            54  24 A5    D0  0CCA  2859          MOVL    UCB$L_CRB(R5),R4                ; Get CRB address
                                     0CCE  2860          ;
```

```
                                0CCE  2861          ; For uVAX I, the max receives and max transmits are fixed,
                                0CCE  2862          ; and there are no map registers on uVAX I!
                                0CCE  2863          ;
                                0CCE  2864          $PUDISP <<790,30$>,-
                                0CCE  2865                  <780,30$>,-
                                0CCE  2866                  <750,30$>,-
                                0CCE  2867                  <730,30$>,-
                                0CCE  2868                  <UV1,70$>>
                                0CE8  2869
                                0CE8  2870 30$:     ; All except uVAX I
          0132 C5    07    90   0CE8  2871          MOVB    #MAX_RCV,UCB$B_XD_RCV_MAX(R5)   ; Set max concurrent receives
          0133 C5    07    90   0CED  2872          MOVB    #MAX_XMT,UCB$B_XD_XMT_MAX(R5)   ; Set max concurrent transmits
                                0CF2  2873 ;
                                0CF2  2874 ; Allocate map registers for recieve buffers.
                                0CF2  2875 ; The unbuffered datapath (DP0) is used for all I/O's due to the fact that
                                0CF2  2876 ; the controller can initiate retransmissions but on the 11/780, the datapath
                                0CF2  2877 ; requires purging before it can be reused.
                                0CF2  2878 ;
        7E A5    42 A5    B0   0CF2  2879          MOVW    UCB$W_DEVBUFSIZ(R5),UCB$W_BCNT(R5) ; Set buffer size
      7C A5    01FF 8F    B0   0CF7  2880          MOVW    #511,UCB$W_BOFF(R5)             ; Set worst case byte offset
                                0CFD  2881          ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                                0CFD  2882          ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
                 34 A4    D4   0CFD  2883          CLRL    CRB$L_INTD+VEC$W_MAPREG(R4)     ; Clear map register + datapath
                                0D00  2884          PUSHQ   R6                              ; Save R6,R7
          56   00C4 C5    DE   0D03  2885          MOVAL   UCB$L_XD_RCV_MAP(R5),R6         ; Get mapping slot address
          57   0132 C5    9A   0D08  2886          MOVZBL  UCB$B_XD_RCV_MAX(R5),R7         ; Get number of rcv map slots
           00000000'GF    16   0D0D  2887 40$:     JSB     G^IOC$ALOUBAMAP                 ; Allocate a set of map registers
                 07 50    E9   0D13  2888          BLBC    R0,50$                          ; Br if unavailable
              86   34 A4    D0   0D16  2889          MOVL    CRB$L_INTD+VEC$W_MAPREG(R4),(R6)+ ; Save receive map info
                 F0 57    F5   0D1A  2890          SOBGTR  R7,40$                          ; Br if more slots available
                                0D1D  2891
                                0D1D  2892 50$:     POPQ    R6                              ; Restore R6,R7
                 13 50    E8   0D20  2893          BLBS    R0,90$                          ; Br if okay so far
          50   0344 8F    3C   0D23  2894          MOVZWL  #SS$_INSFMAPREG,R0              ; Set insufficient map registers
                       05    0D28  2895          RSB                                     ; Return with error
                                0D29  2896
                                0D29  2897 70$:     ; UV1
          0132 C5    04    90   0D29  2898          MOVB    #MAX_RCV_UV1,UCB$B_XD_RCV_MAX(R5) ; Set max concurrent receives
          0133 C5    02    90   0D2E  2899          MOVB    #MAX_XMT_UV1,UCB$B_XD_XMT_MAX(R5) ; Set max concurrent transmits
               50    01    9A   0D33  2900          MOVZBL  S^#SS$_NORMAL,R0                ; Success
                                0D36  2901
                                0D36  2902 90$:     ;
                                0D36  2903          ; Common code path
                                0D36  2904          ;
                 42 A5    A7   0D36  2905          DIVW3   UCB$W_DEVBUFSIZ(R5),-           ; Compute quota for rcv buffers
          51   0136 C5         0D39  2906                  UCB$W_XD_QUOTA(R5),R1           ;
          0132 C5    51    91   0D3D  2907          CMPB    R1,UCB$B_XD_RCV_MAX(R5)         ; Is number less than max?
                       05    1E   0D42  2908          BGEQU   99$                             ; Br if not - leave max alone
          0132 C5    51    90   0D44  2909          MOVB    R1,UCB$B_XD_RCV_MAX(R5)         ; Else, reduce max rcvs allowed
                                0D49  2910 99$:
                                0D49  2911
                                0D49  2912 ;  Now the DMP11 device is master cleared.  A timeout is setup to occur
                                0D49  2913 ;  in approximately one to two seconds.  That should give the DMP enough
                                0D49  2914 ;  time to run the micro-diagnostics. After the timeout, we will procede
                                0D49  2915 ;  and initialize the device.
                                0D49  2916 ;
                                0D49  2917 ;  We will use the WFIKPCH macro which will wait for interrupt or timeout,
```

```
                        0D49  2918 ;  but the interrupt should never occur - only the timeout!  It was a handy
                        0D49  2919 ;  way to schedule a 2 second timeout without a lot of fuss and bother.
                        0D49  2920 ;
                        0D49  2921          ASSUME  IDB$L_CSR EQ 0
     54   2C B4   D0   0D49  2922          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4    ; Get CSR address
                        0D4D  2923          DSBINT  UCB$B_DIPL(R5)                  ; Disable device interrupts
                        0D54  2924
01 A4   40 8F   90   0D54  2925          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)      ; Set the master clear bit
                        0D59  2926 ;
                        0D59  2927 ; WFIKPCH does not destroy R0, therefore R0 should still have SS$_NORMAL
                        0D59  2928 ; success return status.
                        0D59  2929 ;
                        0D59  2930          WFIKPCH START_CONT,#2                   ; Wait 2 seconds for master clr
```

```
                                                J 7
                     0D63  2932          .SBTTL  START_CONT,  Continue after timeout
                     0D63  2933
                     0D63  2934 ;++
                     0D63  2935 ; START_CONT - Continue after timeout
                     0D63  2936 ;
                     0D63  2937 ; Functional description:
                     0D63  2938 ;
                     0D63  2939 ; This routine is called by the system after the startup timeout was set
                     0D63  2940 ; when the device was master cleared.
                     0D63  2941 ;
                     0D63  2942 ;
                     0D63  2943 ; Inputs:
                     0D63  2944 ;
                     0D63  2945 ;       R3 = IRP address
                     0D63  2946 ;       R4 = CSR address
                     0D63  2947 ;       R5 = UCB address
                     0D63  2948 ;
                     0D63  2949 ;       IPL = DIPL
                     0D63  2950 ;
                     0D63  2951 ; Outputs:
                     0D63  2952 ;
                     0D63  2953 ;       R5 is preserved.
                     0D63  2954 ;
                     0D63  2955 ;--
                     0D63  2956
                     0D63  2957 START_CONT:                                  ; Continue with startup
                     0D63  2958          CLRBIT  #UCB$V_TIMOUT,UCB$W_STS(R5) ; Clear timeout status
 0C A5  000018CC'EF   9E  0D68  2959          MOVAB   FORK_PROC,UCB$L_FPC(R5) ; Set fork process PC
       01 A4    80 8F  93  0D70  2960          BITB    #XD_BSEL1_M_RUN,BSEL1(R4) ; Is run bit on?
                07  13  0D75  2961          BEQL    10$                     ; Br if not - error
       06 A4    C5 8F  91  0D77  2962          CMPB    #^0305,BSEL6(R4)        ; Successful master clear?
                15  13  0D7C  2963          BEQL    20$                     ; Yes, continue
 47 A5    06 A4  90  0D7E  2964 10$:     MOVB    BSEL6(R4),UCB$L_DEVDEPEND+3(R5) ; Save BSEL6 on error
  00B8 D5    63  0E  0D83  2965          INSQUE  (R3),@UCB$Q_XD_POST+4(R5) ; Insert IRP on posting Q
  53   01    15  78  0D88  2966          ASHL    #XD_BSEL2_V_ERR+16,#1,R3 ; Indicate fatal error
  54   03    10  78  0D8C  2967          ASHL    #16,#3,R4               ; Indicate device error
          0B10  31  0D90  2968          BRW     SCHED_FORK              ; Request complete
                    0D93  2969
                    0D93  2970 ;
                    0D93  2971 ; The device seems to be running - give mode definition
                    0D93  2972 ;
    04 A4    1B  91  0D93  2973 20$:     CMPB    #^033,BSEL4(R4)         ; Is this a DMV11?
                09  12  0D97  2974          BNEQ    30$                     ; Br if not
    41 A5    17  90  0D99  2975          MOVB    #DT$_DMV11,UCB$B_DEVTYPE(R5) ; Else, set proper device type
  F3AF CF    10  B0  0D9D  2976          MOVW    #MAX_DMV_TRIB,MAX_W_TRIB ; Set maximum number of tribs
                    0DA2  2977 30$:     SETBIT  #UCB$V_XD_INITED,UCB$W_DEVSTS(R5) ; Indicate device inited
                    0DA7  2978          CLRBIT  #UCB$V_XD_INITING,UCB$W_DEVSTS(R5) ; No longer initing
        64    10  88  0DAC  2979          BISB    #XD_BSEL0_M_IEO,BSEL0(R4) ; Enable output interrupts
              01  E0  0DAF  2980          BBS     #XM$V_CHR_LOOPB,-       ; Br if loopback
     03 44 A5          0DB1  2981                  UCB$L_DEVDEPEND(R5),35$ ;
          0083  31  0DB4  2982          BRW     90$                     ; Continue
                    0DB7  2983 35$:
                    0DB7  2984          $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,- ; Dispatch on device type
                    0DB7  2985                  <-      ;function       action
                    0DB7  2986
                    0DB7  2987                          <DT$_DMV11, 40$>,-      ; DMV11
                    0DB7  2988                          <DT$_DMP11, 70$>,-      ; DMP11
```

```
                     0DB7  2989                >
                     0DDA  2990
                     0DDA  2991          ; DMV-11
                     0DDA  2992
          41 8F  90  0DDA  2993 40$:     MOVB    #XD_BSEL1_M_MCLR!XD_BSEL1_M_MAINT,- ; Request Maint mode for DMV
          01 A4      0DDD  2994                  BSEL1(R4)
                     0DDF  2995          TIMEWAIT #9,#XD_BSEL2_M_RDO,SEL2(R4),B  ; Wait for MAINT RDY to be set
                     0E04  2996                                                  ; ... same as RDO bit
          2C 50  E9  0E04  2997          BLBC    R0,50$                          ; Br if ERROR
    02 A4    06  B0  0E07  2998          MOVW    #6,SEL2(R4)                     ; Load LOOP request and clear
                     0E0B  2999                                                  ; "maint ready" bit
                     0E0B  3000          TIMEWAIT #9,#XD_BSEL1_M_RUN,BSEL1(R4),B ; Wait for MAINT RDY to be set
          07 50  E8  0E30  3001          BLBS    R0,90$                          ; Br if success
           FF48  31  0E33  3002 50$:     BRW     10$                             ; Else, process error
                     0E36  3003
                     0E36  3004          ; DMP-11
                     0E36  3005
    01 A4    08  88  0E36  3006 70$:     BISB    #XD_BSEL1_M_LOOPB,BSEL1(R4)     ; Else, enable device loopback
                     0E3A  3007
  0094 C5    63  0E  0E3A  3008 90$:     INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of input Q
              018E  31  0E3F  3009        BRW     LOAD_PORT                       ; Request input port
```

```
                        0E42  3011          .SBTTL  START_TRIB,  Start tributary routine
                        0E42  3012
                        0E42  3013 ;++
                        0E42  3014 ; START_TRIB - Start tributary routine
                        0E42  3015 ;
                        0E42  3016 ; Functional description:
                        0E42  3017 ;
                        0E42  3018 ; This routine is called when a tributary is to be established and started.
                        0E42  3019 ; The polling parameters are initialized also.
                        0E42  3020 ;
                        0E42  3021 ;  Inputs:
                        0E42  3022 ;
                        0E42  3023 ;       R3 = IRP address
                        0E42  3024 ;       R5 = UCB address
                        0E42  3025 ;       R9 = CDB address
                        0E42  3026 ;
                        0E42  3027 ;
                        0E42  3028 ;       IPL = FIPL
                        0E42  3029 ;
                        0E42  3030 ;  Outputs:
                        0E42  3031 ;
                        0E42  3032 ;       R5 is preserved.
                        0E42  3033 ;
                        0E42  3034 ;--
                        0E42  3035
                        0E42  3036 START_TRIB:                                  ; Start tributary
08 0C A9   00   E1      0E42  3037          BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$ ; Br if trib not active
   50   02C4 8F   3C    0E47  3038          MOVZWL  #SS$_DEVACTIVE,R0           ; Else, trib already active
           0D81   31    0E4C  3039          BRW     IO_DONE                     ; Request complete
                        0E4F  3040
                        0E4F  3041 10$:     SETIPL  UCB$B_DIPL(R5)              ; Sync access to device
0098 D5    63   0E      0E53  3042          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5) ; Insert at end of input Q
         0175   31      0E58  3043          BRW     LOAD_PORT                   ; Request the CSR
                        0E5B  3044                                              ; Let return to IOC$INITIATE
                        0E5B  3045                                              ; ..clean up IPL
```

```
                        0E5B  3047          .SBTTL  DEV_ACTIVE,  Device must be active
                        0E5B  3048
                        0E5B  3049 ;++
                        0E5B  3050 ; DEV_ACTIVE - Device must be active
                        0E5B  3051 ;
                        0E5B  3052 ; Functional description:
                        0E5B  3053 ;
                        0E5B  3054 ; This routine is called to queue a request to the device input queue.
                        0E5B  3055 ; The routine must first verify that the device is active, before the
                        0E5B  3056 ; request can be queued, however. If the device is not active, the request
                        0E5B  3057 ; is posted in error.
                        0E5B  3058 ;
                        0E5B  3059 ; Inputs:
                        0E5B  3060 ;
                        0E5B  3061 ;       R3 = IRP address
                        0E5B  3062 ;       R5 = UCB address
                        0E5B  3063 ;
                        0E5B  3064 ;       IPL = FIPL
                        0E5B  3065 ;
                        0E5B  3066 ; Outputs:
                        0E5B  3067 ;
                        0E5B  3068 ;       None.
                        0E5B  3069 ;
                        0E5B  3070 ;--
                        0E5B  3071
                        0E5B  3072 DEV_ACTIVE:                                  ; Check if device is active
08 68 A5    00    E0    0E5B  3073          BBS     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),10$ ; Br if device inited
 50    20D4 8F    3C    0E60  3074          MOVZWL  #SS$_DEVINACT,R0                ; Else, return error
          0D68    31    0E65  3075          BRW     IO_DONE                         ; Finish I/O
                        0E68  3076
                        0E68  3077 10$:     SETIPL  UCB$B_DIPL(R5)                  ; Sync access to device
0098 D5     63    0E    0E6C  3078          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)     ; Insert at end of Q
          015C    31    0E71  3079          BRW     LOAD_PORT                       ; Give request to device
                        0E74  3080                                                  ; Let return to IOC$INITIATE
                        0E74  3081                                                  ; ..clean up IPL
```

```
                        0E74  3083          .SBTTL  TRIB_ACTIVE,  Tributary must be active
                        0E74  3084
                        0E74  3085 ;++
                        0E74  3086 ; TRIB_ACTIVE - Tributary must be active
                        0E74  3087 ;
                        0E74  3088 ; Functional description:
                        0E74  3089 ;
                        0E74  3090 ; This routine is called to queue a request to the device for a tributary.
                        0E74  3091 ; The routine must first verify that the trib is active and if so, queue
                        0E74  3092 ; the request, else it must post the request in error.
                        0E74  3093 ;
                        0E74  3094 ; Inputs:
                        0E74  3095 ;
                        0E74  3096 ;       R3 = IRP address
                        0E74  3097 ;       R5 = UCB address
                        0E74  3098 ;       R9 = CDB address
                        0E74  3099 ;
                        0E74  3100 ;       IPL = FIPL
                        0E74  3101 ;
                        0E74  3102 ; Outputs:
                        0E74  3103 ;
                        0E74  3104 ;       None.
                        0E74  3105 ;
                        0E74  3106 ;--
                        0E74  3107
                        0E74  3108 TRIB_ACTIVE:                                     ; Check if trib is active
08 0C A9   00     E0    0E74  3109          BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$ ; Br if trib established
 50   20D4 8F     3C    0E79  3110          MOVZWL  #SS$_DEVINACT,R0                 ; Set error return
         0D4F     31    0E7E  3111          BRW     IO_DONE                          ; Finish the request
                        0E81  3112
                        0E81  3113 10$:     SETIPL  UCB$B_DIPL(R5)                   ; Sync access to device
0098 D5    63     0E    0E85  3114          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)      ; Insert at end of Q
         0143     31    0E8A  3115          BRW     LOAD_PORT                        ; Load CSR's
                        0E8D  3116                                                   ; Let return to IOC$INITIATE
                        0E8D  3117                                                   ; ..clean up IPL
```

```
                                0E8D  3119          .SBTTL  TRIB_ERRS,  Read trib errors routine
                                0E8D  3120
                                0E8D  3121 ;++
                                0E8D  3122 ; TRIB_ERRS - Read trib errors routine
                                0E8D  3123 ;
                                0E8D  3124 ; Functional description:
                                0E8D  3125 ;
                                0E8D  3126 ; This routine is called to read the tributary error counters.  All
                                0E8D  3127 ; counters are read from the device except for data messages and bytes
                                0E8D  3128 ; transmitted and received.  These counters are 32 bits in length and
                                0E8D  3129 ; are kept by the driver itself.
                                0E8D  3130 ;
                                0E8D  3131 ; Inputs:
                                0E8D  3132 ;
                                0E8D  3133 ;       R3 = IRP address
                                0E8D  3134 ;       R5 = UCB address
                                0E8D  3135 ;       R9 = CDB address
                                0E8D  3136 ;
                                0E8D  3137 ;       IPL = FIPL
                                0E8D  3138 ;
                                0E8D  3139 ; Outputs:
                                0E8D  3140 ;
                                0E8D  3141 ;       The error counters are returned and the request is completed.
                                0E8D  3142 ;
                                0E8D  3143 ;--
                                0E8D  3144
                                0E8D  3145 TRIB_ERRS:
08 0C A9    00    E0            0E8D  3146          BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$ ; Br if trib established
 50   20D4 8F    3C            0E92  3147          MOVZWL  #SS$_DEVINACT,R0                 ; Set error return
         0D36    31            0E97  3148          BRW     IO_DONE                          ; Finish the request
                               0E9A  3149
   52   2C A3    D0            0E9A  3150 10$:     MOVL    IRP$L_SVAPTE(R3),R2              ; Get system P2 address
             2D    13            0E9E  3151          BEQL    20$                              ; Br if none
50   F2B3 CF    9E            0EA0  3152          MOVAB   TRIB_COUNTER,R0                  ; Get address of counter codes
     51   62    D0            0EA5  3153          MOVL    (R2),R1                          ; Get address of data
                               0EA8  3154          PUSHQ   R2                               ; Save P2 buffer address + IRP
   52   38 A9    DE            0EAB  3155          MOVAL   CDB_L_CNTS(R9),R2                ; Get address of counts
                               0EAF  3156          ASSUME  CDB_C_CNTS EQ 16                 ; Assume 16 bytes of XD counts
     53   04    3C            0EAF  3157          MOVZWL  S^#4,R3                          ; Set number of counters
                               0EB2  3158
     81   80    B0            0EB2  3159 15$:     MOVW    (R0)+,(R1)+                      ; Set next code
     81   82    D0            0EB5  3160          MOVL    (R2)+,(R1)+                      ; Set next value
       F7 53    F5            0EB8  3161          SOBGTR  R3,15$                           ; Loop if more
                               0EBB  3162
                               0EBB  3163          POPQ    R2                               ; Restore P2 buffer address + IRP
     62   51    D0            0EBE  3164          MOVL    R1,(R2)                          ; Reset address of next byte
 21 A3    09    91            0EC1  3165          CMPB    #XD_FC_V_RCTERR,IRP$B_XDFUNC(R3) ; Is this a clear counter?
             06    12            0EC5  3166          BNEQ    20$                              ; Br if not
       38 A9    7C            0EC7  3167          CLRQ    CDB_L_CNTS(R9)                   ; Else, clear counters
       40 A9    7C            0ECA  3168          CLRQ    CDB_L_CNTS+8(R9)                 ;       ...
                               0ECD  3169
                               0ECD  3170 20$:     SETIPL  UCB$B_DIPL(R5)                   ; Sync access to device
0098 D5    63    0E            0ED1  3171          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)      ; Insert at end of Q
         00F7    31            0ED6  3172          BRW     LOAD_PORT                        ; Give to device
                               0ED9  3173                                                   ; Let return to IOC$INITIATE
                               0ED9  3174                                                   ; ..clean up IPL
```

```
                        0ED9  3176          .SBTTL  FILLRCVLIST, Fill receive buffer list
                        0ED9  3177          .SBTTL  ADDRCVLIST, Add a buffer to the receive list
                        0ED9  3178
                        0ED9  3179 ;++
                        0ED9  3180 ; FILLRCVLIST - Fill receive buffer list
                        0ED9  3181 ; ADDRCVLIST - Add a buffer to the receive list
                        0ED9  3182 ;
                        0ED9  3183 ; Functional description:
                        0ED9  3184 ;
                        0ED9  3185 ; This routine is entered to make sure that the receive buffer pool
                        0ED9  3186 ; is full.  If it is not, buffers are allocated and queued to the
                        0ED9  3187 ; list until it is.
                        0ED9  3188 ;
                        0ED9  3189 ; Inputs:
                        0ED9  3190 ;
                        0ED9  3191 ;       R2 = Buffer address (ADDRCVLIST only)
                        0ED9  3192 ;       R5 = UCB address
                        0ED9  3193 ;
                        0ED9  3194 ;       IPL = FIPL
                        0ED9  3195 ;
                        0ED9  3196 ; Outputs:
                        0ED9  3197 ;
                        0ED9  3198 ;       R0-R2 are destroyed.
                        0ED9  3199 ;
                        0ED9  3200 ;--
                        0ED9  3201          .ENABL  LSB
                        0ED9  3202 FILLRCVLIST:                                 ; Fill receive buffer list
          00    E1      0ED9  3203          BBC     #UCB$V_XD_INITED,-          ; Exit if not inited
60 68 A5                0EDB  3204                  UCB$W_DEVSTS(R5),40$        ;
          52    D4      0EDE  3205          CLRL    R2                          ; No buffer here
                        0EE0  3206
                        0EE0  3207 ADDRCVLIST:                                  ; Add a buffer to the receive list
                        0EE0  3208          PUSHQ   R3                          ; Save R3, R4
        42 A5   B1      0EE3  3209 5$:      CMPW    UCB$W_DEVBUFSIZ(R5),-       ; Can new block be allocated?
      0136 C5           0EE6  3210                  UCB$W_XD_QUOTA(R5)          ;
            34    1A    0EE9  3211          BGTRU   20$                         ; Br if no - list filled
            51    D4    0EEB  3212          CLRL    R1                          ; Zero size
      004C  8F    A1    0EED  3213          ADDW3   #RCV_T_DATA+CXB$C_TRAILER,- ; Compute block size needed
51    42 A5             0EF1  3214                  UCB$W_DEVBUFSIZ(R5),R1      ;
            52    D5    0EF4  3215          TSTL    R2                          ; Buffer already allocated?
            09    12    0EF6  3216          BNEQ    7$                          ; Br if yes
00000000'GF     16      0EF8  3217          JSB     G^EXE$ALONONPAGED           ; Allocate nonpaged memory
          17    50    E9    0EFE  3218          BLBC    R0,10$                      ; Br if failure
08 A2     51    B0    0F01  3219 7$:      MOVW    R1,RCV_W_BLKSIZE(R2)        ; Insert block size
0A A2     17    90    0F05  3220          MOVB    S^#DYN$C_NET,RCV_B_BLKTYPE(R2) ; Insert block type
00BC C5   62    0E    0F09  3221          INSQUE  (R2),UCB$Q_XD_RCV_BUF(R5)   ; Insert block on free receive list
        42 A5   A2    0F0E  3222          SUBW    UCB$W_DEVBUFSIZ(R5),-       ; Decrement quota
      0136 C5           0F11  3223                  UCB$W_XD_QUOTA(R5)          ;
            52    D4    0F14  3224          CLRL    R2                          ; No more buffers given
            CB    11    0F16  3225          BRB     5$                          ; Try for more
                        0F18  3226 ;
                        0F18  3227 ; Buffer allocation failure
                        0F18  3228 ;
                        0F18  3229 10$:     SETBIT  #XM$V_STS_BUFFAIL,-         ; Set buffer alloc failure
                        0F18  3230                  UCB$L_DEVDEPEND(R5)         ;
            10    11    0F1D  3231          BRB     30$                         ; And give any receives to device
                        0F1F  3232
```

```
                     0F1F  3233 20$:    CLRBIT  #XM$V_STS_BUFFAIL,-                ; Clear buffer alloc failure
                     0F1F  3234                 UCB$L_DEVDEPEND(R5)                ;
          50   52    D0  0F24  3235             MOVL    R2,R0                      ; Get address of buffer
               06    13  0F27  3236             BEQL    30$                        ; Br if none
00000000'GF         16  0F29  3237             JSB     G^COM$DRVDEALMEM           ; Deallocate it
                     0F2F  3238
                     0F2F  3239 30$:    DSBINT  UCB$B_DIPL(R5)                     ; Sync access to device
               07    10  0F36  3240             BSBB    START_RECEIVE              ; Start the receives
                     0F38  3241             ENBINT                             ; Restore previous IPL
                     0F3B  3242             POPQ    R3                         ; Restore R3, R4
               05    05  0F3E  3243 40$:    RSB                                ; Return
                     0F3F  3244             .DSABL  LSB
```

```
                                0F3F  3246          .SBTTL  START_RECEIVE,  Start any receive requests pending
                                0F3F  3247
                                0F3F  3248 ;++
                                0F3F  3249 ; START_RECEIVE - Start any receive requests
                                0F3F  3250 ;
                                0F3F  3251 ; Functional description:
                                0F3F  3252 ;
                                0F3F  3253 ; This routine attemptes to start any receives that may be pending.  This
                                0F3F  3254 ; involves dequeueing a free receive buffer, mapping it, and loading it's
                                0F3F  3255 ; address and size into the device.
                                0F3F  3256 ;
                                0F3F  3257 ; Inputs:
                                0F3F  3258 ;
                                0F3F  3259 ;       R5 = UCB address
                                0F3F  3260 ;
                                0F3F  3261 ;       IPL = DIPL
                                0F3F  3262 ;
                                0F3F  3263 ; Outputs:
                                0F3F  3264 ;
                                0F3F  3265 ;       R5 is preserved.
                                0F3F  3266 ;
                                0F3F  3267 ;       R0-R4 are destroyed
                                0F3F  3268 ;
                                0F3F  3269 ;--
                                0F3F  3270
                                0F3F  3271 START_RECEIVE:                                   ; Start receive operation
          51    0132 C5   9A    0F3F  3272          MOVZBL  UCB$B_XD_RCV_MAX(R5),R1           ; Get max concurrent receives
51  0130 C5   51    00  EB    0F44  3273          FFC     #0,R1,UCB$B_XD_RCV_MAP(R5),R1     ; Get a free mapping slot
                  07    13  0F4B  3274          BEQL    10$                               ; Br if none - just exit
          53    00BC D5   0F  0F4D  3275          REMQUE  @UCB$Q_XD_RCV_BUF(R5),R3          ; Get a free buffer
                  01    1C  0F52  3276          BVC     20$                               ; Br if buffer found
                  05    0F54  3277 10$:     RSB                                           ; Return to caller
                                0F55  3278 ;
                                0F55  3279 ; Mark slot in use and create buffer address/character count image
                                0F55  3280 ; in receive buffer and load UNIBUS adapter map registers.
                                0F55  3281 ;
                                0F55  3282 20$:     SETBIT  R1,UCB$B_XD_RCV_MAP(R5)       ; Mark slot in use
    0B A3     51    90  0F5B  3283          MOVB    R1,RCV_B_MAPSLOT(R3)                  ; Save mapping slot index
        42 A5     B0  0F5F  3284          MOVW    UCB$W_DEVBUFSIZ(R5),-                 ; Insert character count
        0E A3         0F62  3285                  RCV_L_BACC+2(R3)                      ;
                                0F64  3286
                                0F64  3287          CPUDISP <<790,70$>,-
                                0F64  3288                   <780,70$>,-
                                0F64  3289                   <750,70$>,-
                                0F64  3290                   <730,70$>,-
                                0F64  3291                   <UV1,30$>>
                                0F7E  3292
                                0F7E  3293 30$:     ; UV1
 50   00FC C541   D0    0F7E  3294          MOVL    UCB$L_XD_RCV_PA(R5)[R1],R0            ; Get Physical address
    0C A3    50   B0    0F84  3295          MOVW    R0,RCV_L_BACC(R3)                     ; Set BA00-BA15
50   50    F0 8F   78    0F88  3296          ASHL    #-16,R0,R0                            ; Shift down high byte
    10 A3    50   90    0F8D  3297          MOVB    R0,RCV_B_BAHI(R3)                     ; Set BA16-BA21
                  33    11  0F91  3298          BRB     90$                               ; Join common code
                                0F93  3299
                                0F93  3300 70$:     ; All except UV1
 54   00C4 C541   DE    0F93  3301          MOVAL   UCB$L_XD_RCV_MAP(R5)[R1],R4           ; Get mapping info slot address
    51    48 A3   9E    0F99  3302          MOVAB   RCV_T_DATA(R3),R1                     ; Get receive buffer data addr
```

```
                OC A3    51    B0   0F9D  3303          MOVW    R1,RCV_L_BACC(R3)               ; Set BA0-BA8
        0C A3   07   09  64    F0   0FA1  3304          INSV    (R4),#9,#7,RCV_L_BACC(R3)       ; Set BA9-BA15 from map reg
     50 64   06   07     EF   0FA7  3305          EXTZV   #7,#6,(R4),R0                   ; Get BA16-BA17/BA21 also
        0C A3   02   1E  50    F0   0FAC  3306          INSV    R0,#30,#2,RCV_L_BACC(R3)        ; Set BA16-BA17
                        53    DD   0FB2  3307          PUSHL   R3                              ; Save buffer address
                  52  02 A4    3C   0FB4  3308          MOVZWL  2(R4),R2                        ; Set number of map registers
                     53  64    3C   0FB8  3309          MOVZWL  (R4),R3                         ; Set first map register number
                         54    D4   0FBB  3310          CLRL    R4                              ; Use unbuffered datapath
                00000000'GF    16   0FBD  3311          JSB     G^IOC$LOADUBAMAPN               ; Load the map registers
                         53 8ED0   0FC3  3312          POPL    R3                              ; Restore buffer address
                                   0FC6  3313
                                   0FC6  3314 90$:       ;
                                   0FC6  3315            ; Common code path
                                   0FC6  3316            ;
              0094 C5    63    0E   0FC6  3317          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert receive buffer at
                                   0FCB  3318                                                  ;   FRONT of input queue
                         03    10   0FCB  3319          BSBB    LOAD_PORT                       ; Request port and give request
                      FF6F     31   0FCD  3320          BRW     START_RECEIVE                   ; Let's try it again
```

```
                                0FD0  3322          .SBTTL  LOAD_PORT,  Request CSR routine
                                0FD0  3323
                                0FD0  3324 ;++
                                0FD0  3325 ; LOAD_PORT -  Request CSR port routine
                                0FD0  3326 ;
                                0FD0  3327 ; Functional description:
                                0FD0  3328 ;
                                0FD0  3329 ; Request the controller's input port to give it a request.  Since the
                                0FD0  3330 ; controller doesn't service the input port when it is busy, we may have
                                0FD0  3331 ; to request an interrupt to service the request.  The input request is
                                0FD0  3332 ; assume to be on the input waiting queue.
                                0FD0  3333 ;
                                0FD0  3334 ;
                                0FD0  3335 ; Inputs:
                                0FD0  3336 ;
                                0FD0  3337 ;       R5 = UCB address
                                0FD0  3338 ;
                                0FD0  3339 ;       IPL = DIPL
                                0FD0  3340 ;
                                0FD0  3341 ; Outputs:
                                0FD0  3342 ;
                                0FD0  3343 ;       R0 = Success if port loaded immediately
                                0FD0  3344 ;       R1-R3,R9 = destroyed.
                                0FD0  3345 ;       R4 = CSR address
                                0FD0  3346 ;
                                0FD0  3347 ;--
                                0FD0  3348
                                0FD0  3349 LOAD_PORT:                                   ; Give next request to device
54   24 A5   D0                0FD0  3350          MOVL    UCB$L_CRB(R5),R4            ; Get CRB address
                                0FD4  3351          ASSUME  IDB$L_CSR EQ 0
54   2C B4   D0                0FD4  3352          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get CSR address
64   80 8F   93                0FD8  3353          BITB    #XD_BSEL0_M_RQI,(R4)        ; Is an input request pending?
           38   12              0FDC  3354          BNEQ    10$                         ; Br if yes - wait for interrupt
64   80 8F   88                0FDE  3355          BISB    #XD_BSEL0_M_RQI,(R4)        ; Request input port
     53   05   3C              0FE2  3356          MOVZWL  S^#5,R3                     ; Spin loop count
02 A4  80 8F   93              0FE5  3357 5$:      BITB    #XD_BSEL2_M_RDO,BSEL2(R4)   ; Is an output interrupt pend?
           2A   12              0FEA  3358          BNEQ    10$                         ; Br if yes - wait for interrupt
                               0FEC  3359          TIMEWAIT #1,#XD_BSEL2_M_RDI,SEL2(R4),W ; Wait for RDI to be set-10uSEC
        0C 50   E8             1010  3360          BLBS    R0,LOAD_PORT_AVAIL          ; Br if true - port is available
        CF 53   F5             1013  3361          SOBGTR  R3,5$                       ; Loop if more time left
                               1016  3362 ;
                               1016  3363 ; Port is not currently available - request an interrupt to process the input.
                               1016  3364 ;
     64   01   88              1016  3365 10$:     BISB    #XD_BSEL0_M_IEI,(R4)        ; Request input interrupt
          13FC   30            1019  3366          BSBW    SET_INTIM                   ; Start input wait timer
            50   D4            101C  3367          CLRL    R0                          ; Indicate failure to get port
                 05            101E  3368          RSB                                 ; Return to caller
                               101F  3369
                               101F  3370 ;
                               101F  3371 ; Port is available - load request into CSRs
                               101F  3372 ;
                               101F  3373 LOAD_PORT_AVAIL:                             ; Load CSR ports
   64   80 8F   8A             101F  3374          BICB    #XD_BSEL0_M_RQI,(R4)        ; Clear request for port
53   0094 D5   0F              1023  3375          REMQUE  @UCB$Q_XD_INPUTQ(R5),R3     ; Get next request on input Q
           09   1C              1028  3376          BVC     10$                         ; Br if one there
   06 A4   00   B0             102A  3377          MOVW    #XD_CI_V_NOOP,SEL6(R4)      ; Else, release input port
   02 A4   01   B0             102E  3378          MOVW    #XD_I_V_CONTROL,SEL2(R4)    ; Set request type, clear RDI
```

```
                     05  1032  3379          RSB                                     ; Return to caller
                         1033  3380 ;
                         1033  3381 ; Give request from input queue to device
                         1033  3382 ;
          0A   0A A3   91  1033  3383 10$:    CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP     ; Is buffer an IRP ?
                  71   13  1037  3384          BEQL    30$                             ; Br if yes - dispatch
          17   0A A3   91  1039  3385          CMPB    IRP$B_TYPE(R3),S^#DYN$C_NET     ; Is this a receive msg?
                  04   13  103D  3386          BEQL    20$                             ; Br if yes
                         103F  3387          BUG_CHECK NOBUFPCKT,FATAL               ; Else, fatal error
                         1043  3388 ;
                         1043  3389 ; Load receive request
                         1043  3390 ;
                         1043  3391 20$:
                         1043  3393          PUSHQ   R6                              ; Save registers
                    14A5 30  1046  3394          BSBW    XD_BUF_JNX                      ; Get the journal buffer
                  14 50  E9  1049  3395          BLBC    R0,25$                          ; Br if error
                86   08  90  104C  3396          MOVB    #JNL_CSRIN,(R6)+                ; Set journal type = CSRIN
                    86  B4  104F  3397          CLRW    (R6)+                           ; No CHAN
           86   1215 8F  B0  1051  3398          MOVW    #<XD_FC_V_RECV@8>!21,(R6)+      ; Enter FUNC
                    86  7C  1056  3399          CLRQ    (R6)+                           ; STATION = 0
             86   68 A5  B0  1058  3400          MOVW    UCB$W_DEVSTS(R5),(R6)+          ; Set DEVSTS
             86   44 A5  D0  105C  3401          MOVL    UCB$L_DEVDEPEND(R5),(R6)+       ; Set DEVDEPEND
                         1060  3402 25$:    POPQ    R6                              ; Restore registers
                         1063  3404
       04 A4   0C A3   B0  1063  3405          MOVW    RCV_L_BACC(R3),SEL4(R4)         ; Load buffer address
                         1068  3406
                         1068  3407          $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,-
                         1068  3408          <-      ;function       action
                         1068  3409                  <DT$_DMV11, 27$>,-              ; DMV-11
                         1068  3410                  <DT$_DMP11, 28$>,-              ; DMP-11
                         1068  3411          >
                         108B  3412
       06 A4   10 A3   90  108B  3413 27$:    MOVB    RCV_B_BAHI(R3),BSEL6(R4)        ;       ...
       08 A4   0E A3   B0  1090  3414          MOVW    RCV_L_BACC+2(R3),SEL10(R4)      ;  and character count
                   08  B0  1095  3415          MOVW    #XD_I_V_RCV!XD_BSEL2_M_Q22,-    ; Set request type, clear RDI
                02 A4      1097  3416                  SEL2(R4)                        ;
                   09  11  1099  3417          BRB     29$                             ; Continue
                         109B  3418
       06 A4   0E A3   B0  109B  3419 28$:    MOVW    RCV_L_BACC+2(R3),SEL6(R4)       ;  and character count
          02 A4   00   B0  10A0  3420          MOVW    #XD_I_V_RCV,SEL2(R4)            ; Set request type, clear RDI
                         10A4  3421
                         10A4  3422                                                  ;  for trib #0
        00A8 D5   63   0E  10A4  3423 29$:    INSQUE  (R3),@UCB$Q_XD_RCV_PND+4(R5)    ; Insert on end of pending Q
                   05   10A9  3424          RSB                                     ; Return to caller
                         10AA  3425 ;
                         10AA  3426 ; Dispatch to input process handler
                         10AA  3427 ;
                         10AA  3428 30$:
                         10AA  3430          PUSHQ   R6                              ; Save registers
                50   08  90  10AD  3431          MOVB    #JNL_CSRIN,R0                   ; Set journal type = CSRIN
                   141A 30  10B0  3432          BSBW    XD_FILL_JNX                     ; Fill the journal buffer
                  14 50  E9  10B3  3433          BLBC    R0,50$                          ; Br if error
                    66   7C  10B6  3434          CLRQ    (R6)                            ; Clear status
             59   44 A3   D0  10B8  3435          MOVL    IRP$L_CDB(R3),R9                ; Get address of CDB address
                   0C   13  10BC  3436          BEQL    50$                             ; Br if none
                59   69   D0  10BE  3437          MOVL    (R9),R9                         ; Get address of CDB
                   07   13  10C1  3438          BEQL    50$                             ; Br if none
```

```
                                                                ;
                                                                ; Enter CDB status
                                                                ; Enter CDB DEVDEPEND
                                                                ; Restore registers
                                                                ;
                                                                ; Get specific request
                                                                ; Dipatch to input service
                                                                ; Indicate port success
                                                                ;
                                                                ;
                                                                ; Dispatch to input service
                                                                ;
                                                                ;
                                                                ; Start tributary
                                                                ; Start DMP11 controller
                                                                ; Stop tributary
                                                                ; Stop DMP11 controller
                                                                ; Change controller parameters
                                                                ; Change tributary parameters
                                                                ; Read modem register
                                                                ; Read tributary error counts
                                                                ; Read global error counts
                                                                ; Read and clear tributary ctrs
                                                                ; Read and clear global counts
                                                                ; Read TSS location
                                                                ; Read GSS location
                                                                ; Write modem request
                                                                ; Halt tributary
                                                                ; Kill tributary
                                                                ; Xmit request
                                                                ; Halt tributary
                                                                ; Receive request
                                                                ; Set number of sync characters
                                                                ; Latch polling state
                                                                ; Read polling sub-state
                                                                ;
                                                                ;
                                                                ; OTHERS: if not one of the above, then error
                                                                ;
                                                                ; Fatal error
                                                                ;
                                                                ;
                                                                ;
                                                                ; Establish tributary
                                                                ;
                                                                ; Initialize tributary
                                                                ; Get CDB address
                                                                ; Br if CDB found
                                                                ; Else, abort the request
                                                                ;
                                                                ; Br if trib established
                                                                ; and indicate trib established
                                                                ; Establish the trib
                                                                ;
                                                                ; Give trib address
                                                                ; Give request, clear RDI
                                                                ; Insert at front of input Q
```

```
                FEA0   31  112D  3497          BRW     LOAD_PORT                       ; Request input again
                       1130  3498
                       1130  3499 20$:                                                ; Start tributary
      06 A4     03     B0  1130  3500          MOVW    S^#XD_CI_V_ISTRT,SEL6(R4)       ; ISTRT the trib
      08 69     00     E1  1134  3501          BBC     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),30$ ; Br if not MOP mode
      06 A4     04     B0  1138  3502          MOVW    S^#XD_CI_V_MAINT,SEL6(R4)       ; Else, MOP mode
                       113C  3503          SETBIT  #XM$V_STS_RUNNING,-             ; Indicate trib running
                       113C  3504                  CDB_L_DEVDEPEND(R9)             ;
06 A4     0100 8F     A8  1140  3505 30$:     BISW    #XD_SEL6_M_ECOM,SEL6(R4)        ; Enable common buffers
  04 A4     4E A9     90  1146  3506          MOVB    CDB_B_MRB(R9),BSEL4(R4)         ; Set buffer quota
  03 A4     40 A3     90  114B  3507          MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
     02 A4     01     90  1150  3508          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)     ; Give request, clear RDI
                       1154  3509
                       1154  3510          SETBIT  #XM$V_STS_ACTIVE,-              ; Indicate trib active
                       1154  3511                  CDB_L_DEVDEPEND(R9)             ;
            20   A8  1158  3512          BISW    #CD_TS_M_START,-                ; Trib has been started
          0C A9         115A  3513                  CDB_W_STS(R9)                   ;
38 A3     FF 8F     90  115C  3514          MOVB    #255,IRP$L_MEDIA(R3)            ; Set 8 parameters
   21 A3     05     90  1161  3515          MOVB    #XD_FC_V_CHGT,IRP$B_XDFUNC(R3)  ; Set new function request
 0094 C5     63     0E  1165  3516          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of Q
            FE63   31  116A  3517          BRW     LOAD_PORT                       ; Request port again
                       116D  3518
                       116D  3519 ;+
                       116D  3520 ; Define mode
                       116D  3521 ;-
                       116D  3522 IN_INITC:                                        ; Give mode definition
06 A4     02A4 C5     90  116D  3523          MOVB    UCB$B_XD_MODE(R5),BSEL6(R4)     ; Set mode of operation
     02 A4     02     B0  1173  3524          MOVW    #XD_I_V_MODE_DEF,SEL2(R4)       ; Give request, clear RDI
38 A3     F0 8F     90  1177  3525          MOVB    #<15@4>,IRP$L_MEDIA(R3)         ; Set 4 parameters
   21 A3     04     90  117C  3526          MOVB    #XD_FC_V_CHGC,IRP$B_XDFUNC(R3)  ; Set new function request
 0094 C5     63     0E  1180  3527          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of Q
            FE48   31  1185  3528          BRW     LOAD_PORT                       ; Request port again
                       1188  3529 ;+
                       1188  3530 ; Stop tributary
                       1188  3531 ;-
                       1188  3532          .ENABL  LSB
                       1188  3533 IN_STOPT:                                        ; Stop tributary
          0DC5   30   1188  3534          BSBW    GET_CDB                         ; Get CDB address
        03 50    E8   118B  3535          BLBS    R0,10$                          ; Br if CDB present
         0320    31   118E  3536 5$:      BRW     REQ_ABORT                       ; Else, abort the request
                       1191  3537
20 0C A9    03   E4   1191  3538 10$:     BBSC    #CD_TS_V_BUFRET,CDB_W_STS(R9),20$ ; Br if buffers back
            05   E5   1196  3539          BBCC    #CD_TS_V_START,-                ; Br if already halted
     1B 0C A9         1198  3540                  CDB_W_STS(R9),20$               ; ..trib halted itself
   06 A4     05   B0   119B  3541 12$:     MOVW    S^#XD_CI_V_HALT,SEL6(R4)        ; Halt the trib
03 A4    40 A3   90   119F  3542          MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
   02 A4     01   90   11A4  3543          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)     ; Set request, clear RDI
  00002800 8F   CA   11A8  3544          BICL    #XM$M_STS_ACTIVE!XM$M_STS_RUNNING,- ; Indicate trib halted
            69         11AE  3545                  CDB_L_DEVDEPEND(R9)             ;  ...
   34 B9    63   0E   11AF  3546 15$:     INSQUE  (R3),@CDB_Q_INFOUT+4(R9)        ; Wait for buffer returns
         127A   31   11B3  3547          BRW     SET_OUTTIM                      ; Start output wait timer
                       11B6  3548                                                  ; ..and return
                       11B6  3549
            02   E5   11B6  3550 20$:     BBCC    #CD_TS_V_CANCEL,-               ; Br if no CANCEL is pending
     27 0C A9         11B8  3551                  CDB_W_STS(R9),40$               ;
   52    30 A9   9E   11BB  3552          MOVAB   CDB_Q_INFOUT(R9),R2             ; Else, get address of queue
      50    52   D0   11BF  3553          MOVL    R2,R0                           ; Save queue listhead
```

```
                                                K 8
                     52    62    D0  11C2  3554 30$:     MOVL    (R2),R2                         ; Get next IRP
                     50    52    D1  11C5  3555          CMPL    R2,R0                           ; End of queue?
                           E5    13  11C8  3556          BEQL    15$                             ; Br if yes - insert STOP
                  21 A2    10    91  11CA  3557          CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R2) ; CANCEL request?
                           F2    12  11CE  3558          BNEQ    30$                             ; Br if not, keep looking
                     52    62    0F  11D0  3559          REMQUE  (R2),R2                         ; Else, remove CANCEL request
               00B8  D5    62    0E  11D3  3560          INSQUE  (R2),@UCB$Q_XD_POST+4(R5)       ; Post the CANCEL
                  06 A4    00    B0  11D8  3561          MOVW    #XD_CI_V_NOOP,SEL6(R4)          ; Release input port
                  02 A4    01    B0  11DC  3562          MOVW    #XD_I_V_CONTROL,SEL2(R4)        ; Set request type, clear RDI
                           CD    11  11E0  3563          BRB     15$                             ; Let the STOP assume the role
                                     11E2  3564
                           05    E4  11E2  3565 40$:     BBSC    #CD_TS_V_START,-                ; Br if trib started again!
                     B4 0C A9        11E4  3566                  CDB_W_STS(R9),12$               ;
                                     11E7  3567 IN_KILLT:                                        ; Kill trib request
                           00    E1  11E7  3568          BBC     #UCB$V_XD_INITED,-              ; Br if device halted
                        A2 68 A5     11E9  3569                  UCB$W_DEVSTS(R5),5$             ; ..and abort request
                  06 A4    02    B0  11EC  3570          MOVW    S^#XD_CI_V_KILL,SEL6(R4)        ; Set to kill trib
               03 A4    40 A3    90  11F0  3571          MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
                  02 A4    01    90  11F5  3572          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)     ; Give request, clear RDI
                        02C2    31  11F9  3573          BRW     REQ_COM                         ; Request complete
                                     11FC  3574          .DSABL  LSB
                                     11FC  3575 ;+
                                     11FC  3576 ; Stop controller
                                     11FC  3577 ;-
                                     11FC  3578 IN_STOPC:
            13 68 A5    04    E2  11FC  3579          BBSS    #UCB$V_XD_DTRCLR,UCB$W_DEVSTS(R5),10$ ; Br if DTR is clear
                     04 A4    B4  1201  3580          CLRW    SEL4(R4)                        ; Clear DTR
                  06 A4    11    90  1204  3581          MOVB    #XD_CI_V_WTMODEM,BSEL6(R4)      ; Set request type
                  02 A4    01    B0  1208  3582          MOVW    #XD_I_V_CONTROL,SEL2(R4)        ; Give request, clear RDI
               0094 C5    63    0E  120C  3583          INSQUE  (R3),@UCB$Q_XD_INPUTQ(R5)       ; Insert at front of Q
                        FDBC    31  1211  3584          BRW     LOAD_PORT
                                     1214  3585
                                     1214  3586 10$:     CLRBIT  #UCB$V_XD_DTRCLR,UCB$W_DEVSTS(R5) ; Clear DTR flag
                     01 A4    40 8F    90  1219  3587          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)      ; Master clear device
                                     121E  3588
                                     121E  3589 REQ_COM1:                                        ; Long branch to REQ_COM
                           029D    31  121E  3590          BRW     REQ_COM                         ; Request complete
                                     1221  3591 ;+
                                     1221  3592 ; Change controller parameters
                                     1221  3593 ;-
                                     1221  3594 IN_CHGC:                                        ; Change controller parameters
                              00    E1  1221  3595          BBC     #UCB$V_XD_INITED,-              ; Br if device not inited
                           5F 68 A5     1223  3596                  UCB$W_DEVSTS(R5),DEV_INACT1     ; ..to device offline
            51    38 A3    08    00    EA  1226  3597          FFS     #0,#8,IRP$L_MEDIA(R3),R1        ; Find next bit set
                                     122C  3598          CLRBIT  R1,IRP$L_MEDIA(R3)              ; Indicate parameter set
                  04 A4    02A4 C541    B0  1231  3599          MOVW    UCB$W_XD_POLLPRM-<2*4>(R5)[R1],- ; Set parameter value
                                     1238  3600                  SEL4(R4)                        ;  ...
                     06 A4    51    18    A1  1238  3601          ADDW3   #24,R1,SEL6(R4)                 ; Set GSS address
                        06 A4    80 8F    88  123D  3602          BISB    #XD_SEL6_M_WTSS,BSEL6(R4)       ; Request write of GSS
                           02 A4    01    B0  1242  3603          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request, clear RDI
                                 38 A3    95  1246  3604          TSTB    IRP$L_MEDIA(R3)                 ; All done?
                                    08    13  1249  3605          BEQL    20$                             ; If EQL, yes
                              0094 C5    63  124B  3606 10$:     INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of Q
                                    FD7D    31  1250  3607          BRW     LOAD_PORT                       ; Request port again
                                     1253  3608 ;
                                     1253  3609 ; Check if number of sync chars given
                                     1253  3610 ;
```

```
                02A7 C5   95  1253  3611 20$:    TSTB    UCB$B_XD_NMS(R5)                ; Is number of sync chars given
                     C5   13  1257  3612         BEQL    REQ_COM1                        ; Br if not - all done
              21 A3   13   90  1259  3613         MOVB    #XD_FC_V_NUMSYNC,IRP$B_XDFUNC(R3) ; Set new function request
                     EC   11  125D  3614         BRB     10$                             ; Give new request to device
                          125F  3615 ;+
                          125F  3616 ; Set number of sync chars
                          125F  3617 ;-
                          125F  3618 IN_NUMSYNC:
                     00   E1  125F  3619         BBC     #UCB$V_XD_INITED,-              ; Br if device not inited
                 21 68 A5      1261  3620                 UCB$W_DEVSTS(R5),DEV_INACT1     ; ..to device offline
           04 A4   02A7 C5   90  1264  3621         MOVB    UCB$B_XD_NMS(R5),BSEL4(R4)      ; Set parameter value
           06 A4   009B 8F   B0  126A  3622         MOVW    #XD_SEL6_M_WTSS!27,SEL6(R4)     ; Set GSS address + WRITE TSS
              02 A4   01   B0  1270  3623         MOVW    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request, clear RDI
                 0247   31   1274  3624         BRW     REQ_COM                         ; Request complete
                          1277  3625 ;+
                          1277  3626 ; Change tributary
                          1277  3627 ;-
                          1277  3628         .ENABL  LSB
                          1277  3629 IN_CHGT:                                           ; Change controller parameters
                   0CD6   30   1277  3630         BSBW    GET_CDB                         ; Get CDB address
                03 50     E8   127A  3631         BLBS    R0,10$                          ; Br if CDB present
                   0231   31   127D  3632         BRW     REQ_ABORT                       ; Else, abort the request
                          1280  3633
                     00   E0  1280  3634 10$:    BBS     #CD_TS_V_ESTAB,-                ; Br if trib inited
              03 0C A9      1282  3635                 CDB_W_STS(R9),20$               ;       ...
                          1285  3636 DEV_INACT1:                                        ; Long branch to inactive
                   021A   31   1285  3637         BRW     DEV_INACT                       ; Br to device inactive
                          1288  3638
     51   38 A3   08   00   EA  1288  3639 20$:    FFS     #0,#8,IRP$L_MEDIA(R3),R1        ; Find next bit set
                          128E  3640         CLRBIT  R1,IRP$L_MEDIA(R3)              ; Indicate parameter set
        04 A4   50 A941   B0  1293  3641         MOVW    CDB_W_PCLPRM(R9)[R1],SEL4(R4)   ; Set parameter value
        06 A4   51   18   A1  1299  3642         ADDW3   #24,R1,SEL6(R4)                 ; Set TSS address
          06 A4   80 8F   88  129E  3643         BISB    #XD_SEL6_M_WTSS,BSEL6(R4)       ; Request write of TSS
          03 A4   40 A3   90  12A3  3644         MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
             02 A4   01   90  12A8  3645         MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)     ; Set request, clear RDI
                 38 A3   95  12AC  3646         TSTB    IRP$L_MEDIA(R3)                 ; All done?
                     08   13  12AF  3647         BEQL    40$                             ; If EQL, yes
           0094 C5   63   0E  12B1  3648 30$:    INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of Q
                 FD17   31   12B6  3649         BRW     LOAD_PORT                       ; Request port again
                          12B9  3650 ;
                          12B9  3651 ; Set new polling state - if needed else unlatch poll state
                          12B9  3652 ;
              21 A3   14   90  12B9  3653 40$:    MOVB    #XD_FC_V_POLSTAT,IRP$B_XDFUNC(R3) ; Set new request
                     F2   11  12BD  3654         BRB     30$                             ; Ask for port again
                          12BF  3655         .DSABL  LSB
                          12BF  3656 ;+
                          12BF  3657 ; Set new Poll State
                          12BF  3658 ;-
                          12BF  3659 IN_POLSTAT:
                   0C8E   30   12BF  3660         BSBW    GET_CDB                         ; Get CDB address
                03 50     E8   12C2  3661         BLBS    R0,10$                          ; Br if CDB present
                   01E9   31   12C5  3662         BRW     REQ_ABORT                       ; Else, abort the request
                          12C8  3663
                     00   E1  12C8  3664 10$:    BBC     #CD_TS_V_ESTAB,-                ; Br if trib NOT inited
              B8 0C A9      12CA  3665                 CDB_W_STS(R9),DEV_INACT1        ;       ...
                          12CD  3666
                          12CD  3667         $DISPATCH  CDB_B_POL(R9),TYPE=B,<-       ; Dispatch on polling state
```

```
                        12CD  3668                  <NMA$C_CIRPST_AUT  20$>,-       ; AUTOMATIC - unlatch
                        12CD  3669                  <NMA$C_CIRPST_ACT  30$>,-       ; ACTIVE - latch
                        12CD  3670                  <NMA$C_CIRPST_INA  40$>,-       ; INACTIVE - latch
                        12CD  3671                  <NMA$C_CIRPST_DIE  50$>,-       ; DYING - latch
                        12CD  3672                  <NMA$C_CIRPST_DED  60$>>        ; DEAD - latch
                        12DC  3673
                        12DC  3674                                                  ; All others, then
          04 A4   B4    12DC  3675 20$:     CLRW    SEL4(R4)                        ; Clear SEL4
06 A4   1000 8F   B0    12DF  3676          MOVW    #XD_SEL6_M_UPOL,SEL6(R4)        ; Unlatch polling state
             1B   11    12E5  3677          BRB     80$                             ; Continue
                        12E7  3678
          04 A4   94    12E7  3679 30$:     CLRB    BSEL4(R4)                       ; Latch at ACTIVE
                  10    11    12EA  3680          BRB     70$                       ;
       04 A4      01    90    12EC  3681 40$:     MOVB    S^#1,BSEL4(R4)            ; Latch at INACTIVE
                  0A    11    12F0  3682          BRB     70$                       ;
       04 A4      02    90    12F2  3683 50$:     MOVB    S^#2,BSEL4(R4)            ; Latch at DYING
                  04    11    12F6  3684          BRB     70$                       ;
       04 A4      03    90    12F8  3685 60$:     MOVB    S^#3,BSEL4(R4)            ; Latch at DEAD
06 A4    2000 8F   B0    12FC  3686 70$:     MOVW    #XD_SEL6_M_LPOL,SEL6(R4)    ; Latch the poll state
                        1302  3687
  03 A4    40 A3   90    1302  3688 80$:     MOVB    IRP$Q_STATION(R3),BSEL3(R4) ; Set trib address
     02 A4    01   90    1307  3689          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4) ; Set request, clear RDI
           01B0   31    130B  3690          BRW     REQ_COM                     ; Request complete
                        130E  3691 ;+
                        130E  3692 ; Read modem request
                        130E  3693 ;-
                        130E  3694 IN_RDMODEM:                                  ; Read modem register
             00   E0    130E  3695          BBS     #UCB$V_XD_INITED,-          ; Br if device inited
        03 68 A5        1310  3696                  UCB$W_DEVSTS(R5),10$        ;
           018C   31    1313  3697          BRW     DEV_INACT                   ; Else, device inactive
                        1316  3698
       06 A4    10   B0    1316  3699 10$:     MOVW    S^#XD_CI_V_RDMODEM,SEL6(R4) ; Set to read modem register
     02 A4      01   B0    131A  3700          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)  ; Set request, clear RDI
00B0 D5    63   0E    131E  3701          INSQUE  (R3),@UCB$Q_XD_INFOUT+4(R5) ; Insert request at end of Q
         110A   31    1323  3702          BRW     SET_OUTTIM                  ; Set output wait timer
                        1326  3703                                              ; ..and return
                        1326  3704 ;+
                        1326  3705 ; Transmit request
                        1326  3706 ;-
                        1326  3707          .ENABL  LSB
                        1326  3708 IN_XMIT:                                    ; Transmit request
           0C27   30    1326  3709          BSBW    GET_CDB                     ; Get CDB address
        03 50     E8    1329  3710          BLBS    R0,20$                      ; Br if okay
           0182   31    132C  3711 10$:     BRW     REQ_ABORT                   ; And abort the request
                        132F  3712
             0B   E1    132F  3713 20$:     BBC     #XM$V_STS_ACTIVE,-          ; Br if trib not active
        F9 69           1331  3714                  CDB_L_DEVDEPEND(R9),10$     ;
04 A4   38 A3   B0      1333  3715          MOVW    IRP$L_MEDIA(R3),SEL4(R4)    ; Load buffer address
03 A4   40 A3   90      1338  3716          MOVB    IRP$Q_STATION(R3),BSEL3(R4) ; Set trib address
                        133D  3717
                        133D  3718          $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,-
                        133D  3719          <-      ;function       action
                        133D  3720                  <DT$_DMV11, 40$>,-          ; DMV-11
                        133D  3721                  <DT$_DMP11, 60$>,-          ; DMP-11
                        133D  3722          >
                        1360  3723
06 A4   3D A3   90      1360  3724 40$:     MOVB    IRP$L_MEDIA+5(R3),BSEL6(R4) ;       ...
```

```
                08 A4    3A A3    B0  1365  3725          MOVW    IRP$L_MEDIA+2(R3),SEL10(R4)     ;   and character count
                          0C      90  136A  3726          MOVB    S^#XD_I_V_XMIT!XD_BSEL2_M_Q22,- ; Set request, clear RDI
                       02 A4          136C  3727                  BSEL2(R4)
                          09      11  136E  3728          BRB     90$                             ; Continue in common code
                                      1370  3729
                06 A4    3A A3    B0  1370  3730 60$:     MOVW    IRP$L_MEDIA+2(R3),SEL6(R4)      ;   and character count
                   02 A4    04    90  1375  3731          MOVB    S^#XD_I_V_XMIT,BSEL2(R4)        ; Set request, clear RDI
                                      1379  3732
                   2C B9    63    0E  1379  3733 90$:     INSQUE  (R3),@CDB_Q_XMT_PND+4(R9)       ; Insert at end of pending Q
                                05    137D  3734          RSB                                     ;
                                      137E  3735 ;+
                                      137E  3736 ; Receive request
                                      137E  3737 ;-
                                      137E  3738 IN_RECV:                                         ; Bump receive buffer quota
                            0BCF      30  137E  3739          BSBW    GET_CDB                     ; Get CDB address
                          A8 50       E9  1381  3740          BLBC    R0,T0$                      ; Br if error
                             0B       E1  1384  3741          BBC     #XM$V_STS_ACTIVE,-          ; Br if trib not active
                          A4 69           1386  3742                  CDB_L_DEVDEPEND(R9),10$     ;
                   04 A4    01    90  1388  3743          MOVB    S^#T,BSEL4(R4)                  ; Else, bump quota by 1 buffer
                06 A4    0100 8F    B0  138C  3744          MOVW    #XD_SEL6_M_ECOM,SEL6(R4)        ; Set request type
                03 A4    40 A3      90  1392  3745          MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
                   02 A4    01    90  1397  3746          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)     ; Set request, clear RDI
                            0120    31  139B  3747          BRW     REQ_COM
                                      139E  3748          .DSABL  LSB
                                      139E  3749 ;+
                                      139E  3750 ; Read selected tss address
                                      139E  3751 ;-
                                      139E  3752 IN_RDTSS:                                        ; Read tss address
                            0BAF      30  139E  3753          BSBW    GET_CDB                     ; Get CDB address
                          03 50       E8  13A1  3754          BLBS    R0,T0$                      ; Br if got it
                            010A      31  13A4  3755          BRW     REQ_ABORT                   ; Else, abort the request
                                      13A7  3756
                06 A4    3E A3    90  13A7  3757 10$:     MOVB    IRP$L_MEDIA+6(R3),BSEL6(R4)     ; Set tss address
                   06 A4    20    88  13AC  3758          BISB    #XD_SEL6_M_RTSS,BSEL6(R4)       ; Read the tss address
                            2A    11  13B0  3759          BRB     IN_READT                        ; Join common code
                                      13B2  3760 ;+
                                      13B2  3761 ; Read tributary counters
                                      13B2  3762 ;-
                                      13B2  3763          .ENABL  LSB
                                      13B2  3764 IN_RDTERR:                                       ; Read tributary counters
                             7E   20  B0  13B2  3765          MOVW    S^#XD_SEL6_M_RTSS,-(SP)         ; Set to read TSS
                                05    11  13B5  3766          BRB     10$                             ; Continue
                                      13B7  3767
                                      13B7  3768 ;+
                                      13B7  3769 ; Read and clear tributary counters
                                      13B7  3770 ;-
                                      13B7  3771 IN_RCTERR:                                       ; Read and clear trib counters
                          7E   0040 8F    B0  13B7  3772          MOVW    #XD_SEL6_M_RCTS,-(SP)           ; Set to read and clear TSS
                                      13BC  3773
                            0B91      30  13BC  3774 10$:     BSBW    GET_CDB                     ; Get CDB address
                          03 50       E8  13BF  3775          BLBS    R0,T5$                      ; Br if got one
                            00EC      31  13C2  3776          BRW     REQ_ABORT                   ; Else, abort the request
                                      13C5  3777
                               00     E0  13C5  3778 15$:     BBS     #CD_TS_V_ESTAB,-            ; Br if trib established
                          03 0C A9        13C7  3779                  CDB_W_STS(R9),20$           ;
                            00D3      31  13CA  3780          BRW     DEV_INACT_SP                ; Else, device inactive
                                      13CD  3781
```

```
                                1405  3813
                                1405  3814 20$:     BBSC    #CD_TS_V_BUFRET,CDB_W_STS(R9),30$ ; Br if buffers returned
                                140A  3815           BBC     #CD_TS_V_START,-                  ; Br if already halted
                                140C  3816                   CDB_W_STS(R9),10$                 ; ..must be second request
                                140F  3817           BSBB    100$                              ; Else, perform halt on trib
                                1411  3818           BICW    #CD_TS_M_START,-                  ; Indicate trib has
                                1413  3819                   CDB_W_STS(R9)                     ;   been halted
                                1415  3820           BISW    #CD_TS_M_CANCEL,CDB_W_STS(R9)     ; Indicate that we are waiting
                                1419  3821                                                     ;  for $CANCEL to complete
                                1419  3822           INSQUE  (R3),@CDB_Q_INFOUT+4(R9)          ; Wait for buffer returns
                                141D  3823           BRW     SET_OUTTIM                        ; Start output wait timer
                                1420  3824
                                1420  3825 30$:     BICW    #CD_TS_M_CANCEL!CD_TS_M_BUFRET,-; Indicate that $CANCEL
                                1422  3826                   CDB_W_STS(R9)                     ;  completed
                                1424  3827           MOVW    S^#XD_CI_V_ISTRT,SEL6(R4)         ; ISTRT the trib
                                1428  3828           BBC     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),40$ ; Br if not MOP mode
                                142C  3829           MOVW    S^#XD_CI_V_MAINT,SEL6(R4)         ; Else, MOP mode
                                1430  3830           SETBIT  #XM$V_STS_RUNNING,-               ; Indicate trib running
                                1430  3831                   CDB_L_DEVDEPEND(R9)               ;
                                1434  3832 40$:     BISW    #XD_SEL6_M_ECOM,SEL6(R4)          ; Enable common buffers
                                143A  3833           MOVB    CDB_B_MRB(R9),BSEL4(R4)           ; Set buffer quota
                                143F  3834           MOVB    IRP$Q_STATION(R3),BSEL3(R4)       ; Set trib address
                                1444  3835           MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)       ; Give request, clear RDI
                                1448  3836
                                1448  3837           SETBIT  #XM$V_STS_ACTIVE,-                ; Indicate trib active
                                1448  3838                   CDB_L_DEVDEPEND(R9)               ;
```

```
                20   A8  144C  3839          BISW    #CD_TS_M_START,-                ; set Internal flag also
              0C A9       144E  3840                  CDB_W_STS(R9)                   ;
               006B   31  1450  3841 60$:     BRW     REQ_COM                         ; Request complete
                      1453  3842                                                      ; ..and return
                      1453  3843 IN_HALTT:                                           ; Halt trib - no status checks
                  03  10  1453  3844          BSBB    100$                            ; Perform halt on trib
                0066  31  1455  3845          BRW     REQ_COM                         ; Request complete
                      1458  3846
                  00  E1  1458  3847 100$:    BBC     #UCB$V_XD_INITED,-              ; Br if device halted
            54 68 A5        145A  3848                  UCB$W_DEVSTS(R5),REQ_ABORT      ; ..and abort the request
         06 A4    05  90  145D  3849          MOVB    S^#XD_CI_V_HALT,BSEL6(R4)       ; Halt trib request
      03 A4    40 A3  90  1461  3850          MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
         02 A4    01  90  1466  3851          MOVB    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request, clear RDI
                  05  146A  3852          RSB                                     ; Return to caller
                      146B  3853          .DSABL  LSB
                      146B  3854 ;+
                      146B  3855 ; Read selected gss address
                      146B  3856 ;-
                      146B  3857 IN_RDGSS:                                        ; Read gss address
      06 A4    3E A3  90  146B  3858          MOVB    IRP$L_MEDIA+6(R3),BSEL6(R4)     ; Set gss address
         06 A4    20  88  1470  3859          BISB    S^#XD_SEL6_M_RTSS,BSEL6(R4)     ; Request read of gss address
                1E  11  1474  3860          BRB     IN_READG                        ; Finish read gss
                      1476  3861 ;+
                      1476  3862 ; Read global counters
                      1476  3863 ;-
                      1476  3864          .ENABL  LSB
                      1476  3865 IN_RDGERR:                                       ; Read global counters
             7E   20  B0  1476  3866          MOVW    S^#XD_SEL6_M_RTSS,-(SP)         ; Set to read GSS
                  05  11  1479  3867          BRB     10$                             ; Continue
                      147B  3868
                      147B  3869 ;+
                      147B  3870 ; Read and clear global counters
                      147B  3871 ;-
                      147B  3872 IN_RCGERR:                                       ; Read global counters
             7E  0040 8F  B0  147B  3873          MOVW    #XD_SEL6_M_RCTS,-(SP)           ; Set to read and clear GSS
                      1480  3874
                  00  E1  1480  3875 10$:     BBC     #UCB$V_XD_INITED,-              ; Br if device not inited
                1B 68 A5        1482  3876                  UCB$W_DEVSTS(R5),DEV_INACT_SP   ; ..to device inactive
   51   38 A3    08  00  EA  1485  3877          FFS     #0,#8,IRP$L_MEDIA(R3),R1        ; Get next counter index
        06 A4    51  08  A1  148B  3878          ADDW3   #8,R1,SEL6(R4)                  ; Calculate GSS address
              06 A4    8E  A8  1490  3879          BISW    (SP)+,SEL6(R4)                  ; Set read mode
                      1494  3880          .DSABL  LSB
                      1494  3881 IN_READG:                                       ; Finish read gss request
                02 A4    01  B0  1494  3882          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request, clear RDI
                      1498  3883                                                      ;  for trib #0
             00B0 D5    63  0E  1498  3884          INSQUE  (R3),@UCB$Q_XD_INFOUT+4(R5)     ; Insert request at end of Q
                  0F90  31  149D  3885          BRW     SET_OUTTIM                      ; Start output wait timer
                      14A0  3886                                                      ; ..and return
                      14A0  3887 ;+
                      14A0  3888 ; Device inactive
                      14A0  3889 ;-
                      14A0  3890 DEV_INACT_SP:                                    ; Device inactive and pop stack
                    8E  B5  14A0  3891          TSTW    (SP)+                           ; Pop one word from stack
                      14A2  3892 DEV_INACT:                                       ; Device inactive
                      14A2  3893          ; Release the input CSR's
              06 A4    00  B0  14A2  3894          MOVW    S^#XD_CI_V_NOOP,SEL6(R4)        ; Set noop request
              02 A4    01  B0  14A6  3895          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request type, clear RDI
```

```
                50    20D4 8F    B0   14AA  3896          MOVW    #SS$_DEVINACT,R0                ; Set error return code
                           10    11   14AF  3897          BRB     REQ_COM_ERR                     ; Complete request
                                      14B1  3898 ;+
                                      14B1  3899 ; Complete request
                                      14B1  3900 ;-
                                      14B1  3901 REQ_ABORT:                                     ; Abort request
                                      14B1  3902          ; Release the input CSR's
                  06 A4    00    B0   14B1  3903          MOVW    S^#XD_CI_V_NOOP,SEL6(R4)        ; Set noop request
                  02 A4    01    B0   14B5  3904          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request type, clear RDI
                     50    2C    B0   14B9  3905          MOVW    #SS$_ABORT,R0                   ; Set status return
                           03    11   14BC  3906          BRB     REQ_COM_ERR                     ; Complete request in error
                                      14BE  3907 REQ_COM:                                       ; Request complete
                     50    01    B0   14BE  3908          MOVW    S^#SS$_NORMAL,R0                ; Return success
                                      14C1  3909 REQ_COM_ERR:
                  42 A3    50    B0   14C1  3910          MOVW    R0,IRP$W_CSTATUS(R3)            ; Set completion status
              00B8 D5      63    0E   14C5  3911          INSQUE  (R3),@UCB$Q_XD_POST+4(R5)       ; Insert IRP on completion Q
                        03D4    31   14CA  3912          BRW     SCHED_FORKC                     ; Schedule a fork process
```

```
                        14CD  3914          .SBTTL  RDI_INTRPT,  CSR Ready Input Interrupt Service Routine
                        14CD  3915
                        14CD  3916 ;++
                        14CD  3917 ; RDI_INTRPT  -  CSR Ready Input Interrupt Service Routine
                        14CD  3918 ;
                        14CD  3919 ; Functional description:
                        14CD  3920 ;
                        14CD  3921 ; This routine is called when an interrupt occurs to give an input waiting
                        14CD  3922 ; request to the controller.  Prior to execution of this routine, a request
                        14CD  3923 ; was made to LOAD_PORT, but the port was unavailable at that time.
                        14CD  3924 ;
                        14CD  3925 ;
                        14CD  3926 ; Inputs:
                        14CD  3927 ;
                        14CD  3928 ;       00(SP) = address of IDB address
                        14CD  3929 ;       04(SP) - 1C(SP) = R0 - R5
                        14CD  3930 ;
                        14CD  3931 ;       IPL = DIPL
                        14CD  3932 ;
                        14CD  3933 ; Outputs:
                        14CD  3934 ;
                        14CD  3935 ;       The request is loaded and if there are more requests pending,
                        14CD  3936 ;       they are given until there are no more.  Finally, the interrupt
                        14CD  3937 ;       is dismissed.
                        14CD  3938 ;
                        14CD  3939 ;--
                        14CD  3940
                        14CD  3941 RDI_INTRPT:
          54   9E   D0  14CD  3942          MOVL    @(SP)+,R4                       ; Get IDB address
               59   DD  14D0  3943          PUSHL   R9                              ; Save R9
     55   18   A4   D0  14D2  3944          MOVL    IDB$L_UCBLST(R4),R5             ; Get UCB address
          54   64   D0  14D6  3945          MOVL    (R4),R4                         ; Get CSR address
     64   81   8F   8A  14D9  3946          BICB    #XD_BSELO_M_IEI!XD_BSELO_M_RQI,(R4) ; Clear input request flags
               00   E1  14DD  3947          BBC     #UCB$V_XD_INITED,-              ; Exit if controller not inited
        32   68   A5      14DF  3948                  UCB$W_DEVSTS(R5),INTERR         ;
         02A5   C5   94  14E2  3949          CLRB    UCB$B_XD_INTIM(R5)              ; Reset input wait timer
                        14E6  3950 ;
                        14E6  3951 ; Find CDB address
                        14E6  3952 ;
53       0094   D5   0F  14E6  3953          REMQUE  @UCB$Q_XD_INPUTQ(R5),R3         ; Get next request on input Q
               27   1D  14EB  3954          BVS     INTERR                          ; Br if none - error
0094   C5   63   0E  14ED  3955          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Re-insert packet on Q
          FB2A   30  14F2  3956          BSBW    LOAD_PORT_AVAIL                 ; Load CSRs
                        14F5  3957
53       0094   D5   0F  14F5  3958 10$:     REMQUE  @UCB$Q_XD_INPUTQ(R5),R3         ; Get next request on input Q
               0B   1D  14FA  3959          BVS     INTEXIT                         ; Br if none - exit
0094   C5   63   0E  14FC  3960          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Re-insert packet on Q
          FACC   30  1501  3961          BSBW    LOAD_PORT                       ; Load CSRs
        EE   50   F8  1504  3962          BLBS    R0,10$                          ; Br if port was available
                        1507  3963 ;
                        1507  3964 ; Exit interrupt
                        1507  3965 ;
                        1507  3966 INTEXIT:                                       ; Exit interrupt
               59 8ED0  1507  3967          POPL    R9                              ; Restore R9
     50   8E   7D  150A  3968          MOVQ    (SP)+,R0                        ; Restore registers
     52   8E   7D  150D  3969          MOVQ    (SP)+,R2                        ;
     54   8E   7D  1510  3970          MOVQ    (SP)+,R4                        ;
```

```
                02  1513  3971                REI                                     ; Return from interrupt.
                    1514  3972 ;
                    1514  3973 ; An input interrupt occurred with nothing on input queue. Set a noop
                    1514  3974 ; request to device and hope a mode definition has been given previously.
                    1514  3975 ;
                    1514  3976 INTERR:
06 A4   00      90  1514  3977                MOVB    S^#XD_CI_V_NOOP,BSEL6(R4)       ; Set noop request
02 A4   01      B0  1518  3978                MOVW    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request type, clear RDI
        E9      11  151C  3979                BRB     INTEXIT                         ; Exit the interrupt
```

```
                     151E  3981          .SBTTL  RDO_INTRPT,  CSR Ready Output Interrupt Service Routine
                     151E  3982
                     151E  3983 ;++
                     151E  3984 ; RDO_INTRPT - CSR Ready Output Interrupt Service Routine
                     151E  3985 ;
                     151E  3986 ; Functional description:
                     151E  3987 ;
                     151E  3988 ; This routine is called when the DMP11 device generates an output
                     151E  3989 ; interrupt after setting the RDO bit in the BSEL2.  This means that
                     151E  3990 ; the device has some type of information for the driver, solicited
                     151E  3991 ; or otherwise.  This may be buffers being returned from transmits
                     151E  3992 ; or receives, error counters or polling parameters being read, or
                     151E  3993 ; error conditions.
                     151E  3994 ;
                     151E  3995 ;
                     151E  3996 ; Inputs:
                     151E  3997 ;
                     151E  3998 ;       00(SP) = address of IDB address
                     151E  3999 ;       04(SP) - 1C(SP) = R0 - R5
                     151E  4000 ;
                     151E  4001 ;       IPL = DIPL
                     151E  4002 ;
                     151E  4003 ; Outputs:
                     151E  4004 ;
                     151E  4005 ;       Interrupt is dismissed.
                     151E  4006 ;
                     151E  4007 ;--
                     151E  4008
                     151E  4009 RDO_INTRPT:
      54    9E   D0  151E  4010          MOVL    @(SP)+,R4                       ; Get IDB address
   55    18   A4   D0  1521  4011          MOVL    IDB$L_UCBLST(R4),R5             ; Get UCB address
      52    64   D0  1525  4012          MOVL    (R4),R2                         ; Get CSR address
            59   DD  1528  4013          PUSHL   R9                              ; Save R9
            00   E0  152A  4014          BBS     #UCB$V_XD_INITED,-              ; Br if inited
      07 68 A5        152C  4015                  UCB$W_DEVSTS(R5),5$             ;
02 A2    80 8F   8A  152F  4016 3$:       BICB    #XD_BSEL2_M_RDO,BSEL2(R2)       ; Else, release CSRs
            D1   11  1534  4017          BRB     INTEXIT                         ; And exit interrupt
                     1536  4018
   53    02   A2   B0  1536  4019 5$:       MOVW    SEL2(R2),R3                     ; Get SEL2 in R3
      51    53   D0  153A  4020          MOVL    R3,R1                           ; Save type code
53    53    10   78  153D  4021          ASHL    #16,R3,R3                       ; Shift to high word of R3
      53    62   B0  1541  4022          MOVW    SEL0(R2),R3                     ; Get SEL0 in low
                     1544  4023                                                  ;  word of R3
   54    06   A2   B0  1544  4024          MOVW    SEL6(R2),R4                     ; Get SEL6 in high
54    54    10   78  1548  4025          ASHL    #16,R4,R4                       ;  word of R4
   54    04   A2   B0  154C  4026          MOVW    SEL4(R2),R4                     ; Get SEL4 in low
                     1550  4027                                                  ;  word of R4
                     1550  4029          PUSHQ   R6                              ; Save registers
         0C 98   30  1553  4030          BSBW    XD_BUF_JNX                      ; Get the journal buffer
      09 50   E9  1556  4031          BLBC    R0,7$                           ; Br if error
      86    09   90  1559  4032          MOVB    #JNL_CSROUT,(R6)+               ; Set journal type = CSROUT
      86    53   D0  155C  4033          MOVL    R3,(R6)+                        ; Store the CSRs
      86    54   D0  155F  4034          MOVL    R4,(R6)+                        ;       ...
                     1562  4035 7$:       POPQ    R6                              ; Restore registers
                     1565  4037
   50    03   A2   90  1565  4038          MOVB    BSEL3(R2),R0                    ; Get trib address
            59   D4  1569  4039          CLRL    R9                              ; Assume no CDB
```

```
                     50   95  156B  4040          TSTB    R0                              ; Trib #0?
                     06   13  156D  4041          BEQL    10$                             ; Br if yes - no CDB
                   0952   30  156F  4042          BSBW    FIND_TRIB                       ; Find CDB from trib address
                  BA 50   E9  1572  4043          BLBC    R0,3$                           ; Br if trib not found - exit
51    FFFFFFF8 8F    CA  1575  4044 10$:     BICL    #^C<XD_BSEL2_M_TYPE>,R1         ; Clear all but type code
                         157C  4045          CASE    R1,TYPE=B,<OUT_RCV,OUT_CTL,OUT_INFO,OUT_RCV_ERR,- ;
                         157C  4046                  OUT_XMIT,3$,OUT_XMIT_ERR,OUT_XMIT_ERR> ; Case on the type code
                         1590  4047 ;+
                         1590  4048 ; Process receives completed by the DMP11. After the receive buffer is
                         1590  4049 ; posted for completion, any receive buffers that are need are given to
                         1590  4050 ; the device.
                         1590  4051 ;-
                         1590  4052          .ENABL  LSB
                         1590  4053 OUT_RCV:                                            ; Process receive complete
          51   52   D0   1590  4054          MOVL    R2,R1                           ; Save CSR address
     52   00A4 D5   0F   1593  4055          REMQUE  @UCB$Q_XD_RCV_PND(R5),R2        ; Get oldest pending receive
                12   1D   1598  4056          BVS     15$                             ; Error if none.
54    C0000000 8F   CA   159A  4057          BICL    #^XC0000000,R4                  ; Clear BA16 and BA17 from BA/CC
      0C A2    54   B1   15A1  4058          CMPW    R4,RCV_L_BACC(R2)               ; Buffer address match?
                08   13   15A5  4059          BEQL    20$                             ; Br if yes - ok
     00A4 C5    62   0E   15A7  4060 10$:     INSQUE  (R2),UCB$Q_XD_RCV_PND(R5)       ; Requeue the receive buffer
              FF58   31   15AC  4061 15$:     BRW     INTEXIT                         ; Exit the interrupt
      50    0B A2   9A   15AF  4062 20$:     MOVZBL  RCV_B_MAPSLOT(R2),R0            ; Get mapping slot number
 EE 0130 C5    50   E5   15B3  4063          BBCC    R0,UCB$B_XD_RCV_MAP(R5),10$     ; Mark as free - error if not
                         15B9  4064
                         15B9  4065          $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,-
                         15B9  4066          <-      ;function       action
                         15B9  4067                  <DT$_DMV11, 23$>,-              ; DMV-11
                         15B9  4068                  <DT$_DMP11, 26$>,-              ; DMP-11
                         15B9  4069          >
                         15DC  4070
   0E A2    08 A1   B0   15DC  4071 23$:     MOVW    SEL10(R1),RCV_L_BACC+2(R2)      ; Save byte count for recv
                04   11   15E1  4072          BRB     30$                             ; Continue in common code
                         15E3  4073
      0C A2    54   D0   15E3  4074 26$:     MOVL    R4,RCV_L_BACC(R2)               ; Save byte count
                         15E7  4075
   02 A1    80 8F   8A   15E7  4076 30$:     BICB    #XD_BSEL2_M_RDO,SEL2(R1)        ; Release CSRs
      51    0E A2   B0   15EC  4077          MOVW    RCV_L_BACC+2(R2),R1             ; Get byte count
                         15F0  4078          CPUDISP <<790,37$>,-
                         15F0  4079                   <780,37$>,-
                         15F0  4080                   <750,37$>,-
                         15F0  4081                   <730,37$>,-
                         15F0  4082                   <UV1,33$>>
                         160A  4083
   50    0114 C540   D0   160A  4084 33$:     MOVL    UCB$L_XD_RCV_VA(R5)[R0],R0      ; Get buffer VA
                3C   BB   1610  4085          PUSHR   #^M<R2,R3,R4,R5>                ; Save registers
 48 A2    60   51   28   1612  4086          MOVC3   R1,(R0),RCV_T_DATA(R2)          ; Copy the data
                3C   BA   1617  4087          POPR    #^M<R2,R3,R4,R5>                ; Restore registers
53   53   08   18   EF   1619  4088 37$:     EXTZV   #24,#8,R3,R3                    ; Get trib address
                06   13   161E  4089          BEQL    45$                             ; Br if trib zero????
      0C A2    53   90   1620  4090          MOVB    R3,RCV_L_BACC(R2)               ; Return trib address
                59   11   1624  4091          BRB     80$                             ; Complete the rcv in fork
                         1626  4092 ;
                         1626  4093 ; Bogus trib address
                         1626  4094 ;
    00BC C5    62   0E   1626  4095 45$:     INSQUE  (R2),UCB$Q_XD_RCV_BUF(R5)       ; Requeue buffer to device
              FED9   31   162B  4096          BRW     INTEXIT                         ; Exit the interrupt
```

```
                     162E  4097 ;+
                     162E  4098 ; Process a transmit buffer complete.
                     162E  4099 ;-
                     162E  4100 OUT_XMIT:                                       ; Process xmit complete
02 A2    80 8F    8A  162E  4101          BICB    #XD_BSEL2_M_RDO,SEL2(R2)      ; Release CSRs
              59    D5  1633  4102          TSTL    R9                            ; CDB address given?
              17    13  1635  4103          BEQL    50$                           ; Br if not - exit interrupt
   52    28 B9    0F  1637  4104          REMQUE  @CDB_Q_XMT_PND(R9),R2         ; Get next xmit for trib
              11    1D  163B  4105          BVS     50$                           ; Error if none
54   C0000000 8F    CA  163D  4106          BICL    #^XC0000000,R4                ; Clear BA16 and BA17 from BA/CC
   38 A2    54    B1  1644  4107          CMPW    R4,IRP$L_MEDIA(R2)            ; Buffer address match?
              07    13  1648  4108          BEQL    60$                           ; Br if yes - ok
   28 A9    62    0E  164A  4109          INSQUE  (R2),CDB_Q_XMT_PND(R9)        ; Else, requeue the request
         FEB6    31  164E  4110 50$:      BRW     INTEXIT                       ; Exit the interrupt
                     1651  4111
   38 A2    54    D0  1651  4112 60$:      MOVL    R4,IRP$L_IOST1(R2)            ; Save byte count
   42 A2    01    B0  1655  4113          MOVW    S^#SS$_NORMAL,IRP$W_CSTATUS(R2) ; Set success
              24    11  1659  4114          BRB     80$                           ; Queue the xmit for completion
                     165B  4115
                     165B  4116 OUT_XMIT_ERR:                                   ; Xmit not sent
02 A2    80 8F    8A  165B  4117          BICB    #XD_BSEL2_M_RDO,SEL2(R2)      ; Release CSRs
              59    D5  1660  4118          TSTL    R9                            ; CDB address present
              EA    13  1662  4119          BEQL    50$                           ; Br if not - exit
   52    28 B9    0F  1664  4120          REMQUE  @CDB_Q_XMT_PND(R9),R2         ; Get next xmit pending
              E4    1D  1668  4121          BVS     50$                           ; Br if none
              0F    11  166A  4122          BRB     70$                           ;
                     166C  4123
                     166C  4124 OUT_RCV_ERR:                                    ; Receive buffer unused
02 A2    80 8F    8A  166C  4125          BICB    #XD_BSEL2_M_RDO,SEL2(R2)      ; Release CSRs
 52    00A4 D5    0F  1671  4126          REMQUE  @UCB$Q_XD_RCV_PND(R5),R2      ; Get oldest pending receive
              D6    1D  1676  4127          BVS     50$                           ; Br if none
   0C A2    94  1678  4128          CLRB    RCV_L_BACC(R2)                ; Set receive error return
   42 A2    2C    B0  167B  4129 70$:      MOVW    #SS$_ABORT,IRP$W_CSTATUS(R2)  ; Set error return
                     167F  4130
 00B8 D5    62    0E  167F  4131 80$:      INSQUE  (R2),@UCB$Q_XD_POST+4(R5)     ; Queue request for posting
              03    12  1684  4132          BNEQ    90$                           ; Br if not first entry
            0218    30  1686  4133          BSBW    SCHED_FORKC                   ; Schedule fork process
            F8B3    30  1689  4134 90$:      BSBW    START_RECEIVE                 ; Start any receives
            FE78    31  168C  4135          BRW     INTEXIT                       ; Exit interrupt
                     168F  4136          .DSABL  LSB
                     168F  4137
                     168F  4138 ;+
                     168F  4139 ; This routine is called when the device interrupts with a INFORMATION OUT
                     168F  4140 ; command being returned to the driver.  This command results from an
                     168F  4141 ; associated CONTROL IN command, requesting some information from the device.
                     168F  4142 ; Since the information may be of some global type, such as global error
                     168F  4143 ; counters, or the modem register contents; the DMP11 device may not return
                     168F  4144 ; a tributary address on OUTPUT. In this case all global requests are
                     168F  4145 ; assumed to be processed in a FIFO order and the global requests are
                     168F  4146 ; queued to the UCB INFO OUT QUEUE (and not the CDB INFO OUT QUEUE).
                     168F  4147 ;-
                     168F  4148 OUT_INFO:                                       ; Process information out
02 A2    80 8F    8A  168F  4149          BICB    #XD_BSEL2_M_RDO,SEL2(R2)      ; Release CSRs
52   54    08    10  EF  1694  4150          EXTZV   #16,#8,R4,R2                  ; Get BSEL6
   52    60 8F    93  1699  4151          BITB    #<XD_SEL6_M_RTSS!XD_SEL6_M_RCTS>,R2 ; Was this a read request?
              05    12  169D  4152          BNEQ    5$                            ; Br if yes
         52    08    91  169F  4153          CMPB    #8,R2                         ; Was it a read modem request?
```

```
                                                                                ; Br if not
                06   12  16A2  4154              BNEQ    10$
             0123   30  16A4  4155  5$:          BSBW    OUT_STATUS              ; Else, process status
             FE5D   31  16A7  4156              BRW     INTEXIT                 ; Exit interrupt
                        16AA  4157
           52  10   91  16AA  4158  10$:         CMPB    #16,R2                  ; Is it a BUFFER RETURN COMPLETE?
                32   12  16AD  4159              BNEQ    30$                     ; Br if not - exit interrupt
                59   D5  16AF  4160              TSTL    R9                      ; CDB address found?
                2E   13  16B1  4161              BEQL    30$                     ; Br if no - exit interrupt
                         16B3  4162              SETBIT  #CD_TS_V_BUFRET,CDB_W_STS(R9)   ; Note buffers returned
                         16B8  4163 ;
                         16B8  4164 ; We will scan the information out queue for the stop request
                         16B8  4165 ;
           53  30 A9    9E  16B8  4166              MOVAB   CDB_Q_INFOUT(R9),R3     ; Get address
               50   53   D0  16BC  4167              MOVL    R3,R0                   ; Save queue listhead address
               53   63   D0  16BF  4168  20$:       MOVL    (R3),R3                 ; Get next IRP
               50   53   D1  16C2  4169              CMPL    R3,R0                   ; End of queue?
                    1A   13  16C5  4170              BEQL    30$                     ; Br if yes - exit interrupt
           21  A3   10   91  16C7  4171              CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R3) ; Is this a CANCEL request?
                06   13  16CB  4172              BEQL    25$                     ; Br if yes - complete it
           21  A3   02   91  16CD  4173              CMPB    #XD_FC_V_STOPT,IRP$B_XDFUNC(R3) ; Is this the stop request?
                EC   12  16D1  4174              BNEQ    20$                     ; Br if no - keep looking
               53   63   0F  16D3  4175  25$:       REMQUE  IRP$L_IOQFL(R3),R3      ; Else, get the IRP
             0D57   30  16D6  4176              BSBW    SET_OUTTIM              ; Start output wait timer
         0094  C5   63   0E  16D9  4177              INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert stop at front of Q
             FBEF   30  16DE  4178              BSBW    LOAD_PORT               ; Give request to device
             FE23   31  16E1  4179  30$:       BRW     INTEXIT                 ; Else, exit interrupt
                        16E4  4180
                        16E4  4181 ;+
                        16E4  4182 ; This routine is called when the DMP11 device interrupts with a CONTROL
                        16E4  4183 ; OUT command.  Some type of error has occurred and the driver must notify
                        16E4  4184 ; the user and also must take any appropriate steps to handle the error.
                        16E4  4185 ;-
                        16E4  4186 OUT_CTL:                                     ; Process control out
       02  A2     80  8F   8A  16E4  4187              BICB    #XD_BSEL2_M_RDO,SEL2(R2)        ; Release CSRs
 51    54     08      10   EF  16E9  4188              EXTZV   #16,#8,R4,R1            ; Get BSEL6
          51     C0  8F   91  16EE  4189              CMPB    #^O<300>,R1             ; Is this buffer too small?
                     1C   13  16F2  4190              BEQL    PROCEDURE_ERR           ; Br if yes - procedure error
          51     C4  8F   91  16F4  4191              CMPB    #^O<304>,R1             ; Is this a disconnect?
                     05   12  16F8  4192              BNEQ    10$                     ; Br if not
                         16FA  4193              SETBIT  #XM$V_STS_DISC,UCB$L_DEVDEPEND(R5) ; Set error flag
 52    51     02      06   EF  16FF  4194  10$:       EXTZV   #6,#2,R1,R2             ; Else, get severity code
                         1704  4195              CASE    R2,TYPE=B,<-            ; Process by severity
                         1704  4196                      SOFT_ERR,-              ;   Soft error
                         1704  4197                      PROCEDURE_ERR,-         ;   Procedure error
                         1704  4198                      PROCEDURE_ERR,-         ;   Lots of procedure errors
                         1704  4199                      FATAL_ERR>              ;   Fatal error
                         1710  4200 ;+
                         1710  4201 ; PROCEDURE_ERR - Process procedure error from device
                         1710  4202 ;-
                         1710  4203 PROCEDURE_ERR:                              ; Process procedure error
                     59   D5  1710  4204              TSTL    R9                      ; CDB address found?
                     66   13  1712  4205              BEQL    NOOP                    ; Br if no - exit now
                         1714  4206              SETBIT  #XM$V_ERR_TRIB,CDB_L_DEVDEPEND(R9) ; Indicate trib error
                         1718  4207              ;BRB    FORCE_HALT              ; Force trib to shutdown
                         1718  4208
                         1718  4209 FORCE_HALT:                                ; Force trib to shutdown
          0C  A9     20   AA  1718  4210              BICW    #CD_TS_M_START,CDB_W_STS(R9)    ; Clear started flag
```

```
                                171C  4211 FORCE_HALT1:
          53   54    08    10    EF  171C  4212          EXTZV   #16,#8,R4,R3                    ; Get BSEL6 value
                 03 A9    53    90  1721  4213          MOVB    R3,CDB_L_DEVDEPEND+3(R9)        ; Save BSEL6 in CDB
                 00002800 8F    CA  1725  4214          BICL    #XM$M_STS_RUNNING!XM$M_STS_ACTIVE,- ; Tributary is halted
                          69        1728  4215                  CDB_L_DEVDEPEND(R9)             ;
                 53   029C C5    DE  172C  4216          MOVAL   UCB$L_XD_HALT(R5),R3            ; Set address of halted trib list
                                1731  4217 SET_INDEX:                                           ; Set trib index in list
                   52    0E A9    9A  1731  4218          MOVZBL  CDB_B_TRB_ADDR(R9),R2           ; Get trib address from CDB
          0138 C5  EA17 CF    52    3A  1735  4219          LOCC    R2,#MAX_W_TRIB,UCB$B_XD_TRIB_VEC(R5)    ; Find trib address
                          3B    13  173D  4220          BEQL    NOOP                            ; Exit if not found ???
                 52   0138 C5    9E  173F  4221          MOVAB   UCB$B_XD_TRIB_VEC(R5),R2        ; Else, get address of vector
                      51   52    C2  1744  4222          SUBL2   R2,R1                           ; Calculate index
                                1747  4223          SETBIT  R1,(R3)                         ; Set trib index
                                174B  4224 SFORK_EXIT:                                        ; Exit with a fork process
                           0153   30  174B  4225          BSBW    SCHED_FORKC                     ; Schedule a fork process
                             2A   11  174E  4226          BRB     NOOP                            ; Exit the interrupt
                                1750  4227
                                1750  4228 ;+
                                1750  4229 ; SOFT_ERR - Process soft error from device
                                1750  4230 ;-
                                1750  4231 SOFT_ERR:                                           ; Process soft error
                             59   D5  1750  4232          TSTL    R9                              ; CDB address found?
                             26   13  1752  4233          BEQL    NOOP                            ; Br if no - exit now
          52   51    06    00    EF  1754  4234          EXTZV   #0,#6,R1,R2                     ; Get error type
            52   52    FF 8F    78  1759  4235          ASHL    #-1,R2,R2                       ; Convert type to case code
                                175E  4236          CASE    R2,TYPE=B,LIMIT=#1,<-           ; Process by type
                                175E  4237                  THRESH,-                        ;   Receive threshold error
                                175E  4238                  THRESH,-                        ;   Transmit threshold error
                                175E  4239                  THRESH,-                        ;   Select threshold error
                                175E  4240                  START_ERR,-                     ;   Start when running
                                175E  4241                  MAINT_ERR,-                     ;   MOP when running
                                175E  4242                  NOOP,-                          ;   MOP when halted
                                175E  4243                  START_ERR1,-                    ;   Start when in MOP
                                175E  4244                  NOOP,-                          ;   Reserved
                                175E  4245                  DEAD_ERR,-                      ;   Dead tributary
                                175E  4246                  RUNNING_NOTIF,-                 ;   Running tributary
                                175E  4247                  PROCEDURE_ERR,-                 ;   Babbling trib
                                175E  4248                  PROCEDURE_ERR>                  ;   Streaming trib
                                177A  4249 ;
                                177A  4250 ; All others
                                177A  4251 ;
                                177A  4252 NOOP:                                              ; No special operation
        FD8A    31              177A  4253          BRW     INTEXIT                         ; Just exit the interrupt
                                177D  4254
                                177D  4255 THRESH:                                            ; Threshold errors
                                177D  4256          SETBIT  #XM$V_ERR_THRESH,CDB_L_DEVDEPEND(R9) ; Indicate warning
          F7    11              1781  4257          BRB     NOOP                            ; And exit interrupt
                                1783  4258
                                1783  4259 START_ERR:                                        ; Start received in RUN
                                1783  4260          SETBIT  #XM$V_ERR_START,CDB_L_DEVDEPEND(R9) ; Indicate cause of error
          8F    11              1787  4261          BRB     FORCE_HALT                      ; Force trib to shutdown
                                1789  4262
                                1789  4263 START_ERR1:                                       ; Start received in MOP
                                1789  4264          SETBIT  #XM$V_ERR_START,CDB_L_DEVDEPEND(R9) ; Indicate start received
          EB    11              178D  4265          BRB     NOOP                            ; Exit the interrupt
                                178F  4266
                                178F  4267 MAINT_ERR:                                         ; MOP received while running
```

```
                                 178F  4268          SETBIT  #XM$V_ERR_MAINT,CDB_L_DEVDEPEND(R9) ; Indicate cause of error
          0C A9    20    A8      1793  4269          BISW    #CD_TS_M_START,CDB_W_STS(R9)    ; Trib starts up by itself in MOP
                   83    11      1797  4270          BRB     FORCE_RACT1                     ; Force trib shutdown
                                 1799  4271
                                 1799  4272 DEAD_ERR:                                       ; Dead tributary
                                 1799  4273          CLRBIT  #XM$V_STS_RUNNING,CDB_L_DEVDEPEND(R9) ; No longer running
                   DB    11      179D  4274          BRB     NOOP                            ; Exit the interrupt
                                 179F  4275
                                 179F  4276 RUNNING_NOTIF:                                  ; Running tributary
                                 179F  4277          SETBIT  #XM$V_STS_RUNNING,-             ; Indicate trib is running
                                 179F  4278                  CDB_L_DEVDEPEND(R9)             ;
          53    0298 C5    DE      17A3  4279          MOVAL   UCB$L_XD_RUN(R5),R3             ; Set address of run list
                     FF86    31      17A8  4280          BRW     SET_INDEX                       ; Set trib index in run list
                                 17AB  4281
                                 17AB  4282 ;+
                                 17AB  4283 ; FATAL_ERR - Process a fatal device error
                                 17AB  4284 ;-
                                 17AB  4285 FATAL_ERR:                                      ; Process a fatal device error
            52    24 A5    D0      17AB  4286          MOVL    UCB$L_CRB(R5),R2                ; Get CRB address
                                 17AF  4287          ASSUME  IDB$L_CSR EQ 0
            52    2C B2    D0      17AF  4288          MOVL    @CRB$L_INTD+VEC$L_IDB(R2),R2    ; Get CSR address
       01 A2    40 8F    90      17B3  4289          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R2)      ; Master clear the device
53     54    08    10    EF      17B8  4290          EXTZV   #16,#8,R4,R3                    ; Get BSEL6 value
         47 A5    53    90      17BD  4291          MOVB    R3,UCB$L_DEVDEPEND+3(R5)        ; Save error code
          53    01    15    78      17C1  4292          ASHL    #XD_BSEL2_V_ERR+16,#1,R3        ; Note fatal error
                   00DB    30      17C5  4293          BSBW    SCHED_FORK                      ; Schedule a fork process
                     B0    11      17C8  4294          BRB     NOOP                            ; And exit
```

```
                                17CA  4296          .SBTTL  OUT_STATUS,  Process Information Out from device
                                17CA  4297
                                17CA  4298 ;++
                                17CA  4299 ; OUT_STATUS - Process Information Out from device
                                17CA  4300 ;
                                17CA  4301 ; Functional description:
                                17CA  4302 ;
                                17CA  4303 ; This routine get the next request from the information out waiting queue
                                17CA  4304 ; and returns the information from the CSRs.
                                17CA  4305 ;
                                17CA  4306 ; Inputs:
                                17CA  4307 ;
                                17CA  4308 ;       R3 = SEL0 and SEL2
                                17CA  4309 ;       R4 = SEL4 and SEL6
                                17CA  4310 ;       R5 = UCB address
                                17CA  4311 ;       R9 = CDB address (only if tributary information)
                                17CA  4312 ;
                                17CA  4313 ;       IPL = DIPL
                                17CA  4314 ;
                                17CA  4315 ; Outputs:
                                17CA  4316 ;
                                17CA  4317 ;       R0-R3 = Destroyed
                                17CA  4318 ;--
                                17CA  4319
                                17CA  4320 OUT_STATUS:                                  ; Process information out
      51   53   08   18   EF    17CA  4321          EXTZV   #24,#8,R3,R1                ; Get trib address
                     55   13    17CF  4322          BEQL    40$                         ; Br if global request
                                17D1  4323 ;
                                17D1  4324 ; Read from TSS
                                17D1  4325 ;
      52    30 A9    9E         17D1  4326          MOVAB   CDB_Q_INFOUT(R9),R2         ; Get address
         50    52    D0         17D5  4327          MOVL    R2,R0                       ; Save queue listhead address
         52    62    D0         17D8  4328 5$:      MOVL    (R2),R2                     ; Get next IRP
         50    52    D1         17DB  4329          CMPL    R2,R0                       ; End of queue?
               3E    13         17DE  4330          BEQL    20$                         ; Br if yes - exit
                                17E0  4331 ;
                                17E0  4332 ; Only a read TSS, read counter, CANCEL and a stop can be on the INFOUT queue.
                                17E0  4333 ; Find the first non-stop request.
                                17E0  4334 ;
      21 A2    02   91         17E0  4335          CMPB    #XD_FC_V_STOPT,IRP$B_XDFUNC(R2) ; Is this a stop request?
               F2   13         17E4  4336          BEQL    5$                          ; Br if yes - keep looking
      21 A2    10   91         17E6  4337          CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R2) ; Is this a CANCEL request?
               EC   13         17EA  4338          BEQL    5$                          ; Br if yes - keep looking
         52    62   0F         17EC  4339          REMQUE  IRP$L_IOQFL(R2),R2          ; Else get read request
            0C3E    30         17EF  4340          BSBW    SET_OUTTIM                  ; Start output wait timer
      21 A2    15   91         17F2  4341          CMPB    #XD_FC_V_POLSS,IRP$B_XDFUNC(R2) ; Is this the polling substate
               0B   12         17F6  4342          BNEQ    7$                          ; Br if not
54    54    F8   8F   78       17F8  4343          ASHL    #-8,R4,R4                   ; Shift down high byte
      3F A2    54   90         17FD  4344          MOVB    R4,IRP$B_POLS(R2)           ; Save return info
               0D   11         1801  4345          BRB     15$                         ; Continue
      51    2C A2    D0         1803  4346 7$:      MOVL    IRP$L_SVAPTE(R2),R1         ; Get system buffer address
            38 A2    D5         1807  4347          TSTL    IRP$L_MEDIA(R2)             ; Is this a counter request?
               13   12         180A  4348          BNEQ    30$                         ; Br if yes
      0C A1    54   3C         180C  4349 10$:     MOVZWL  R4,P2B_T_DATA(R1)           ; Else, store return info.
      42 A2    01   B0         1810  4350 15$:     MOVW    #SS$_NORMAL,IRP$W_CSTATUS(R2) ; Set success return
   00B8 D5     62   0E         1814  4351          INSQUE  (R2),@UCB$Q_XD_POST+4(R5)   ; Queue request for posting
               03   12         1819  4352          BNEQ    20$                         ; Br if not first entry
```

```
                     0083   30  181B  4353          BSBW    SCHED_FORKC                     ; Schedule fork process
                            05  181E  4354 20$:     RSB                                     ; Return to caller
                                181F  4355 ;
                                181F  4356 ; Trib counters request
                                181F  4357 ;
                50   E93A CF   9E  181F  4358 30$:     MOVAB   TRIB_COUNTER1-2,R0              ; Get address of counter types
                          18   11  1824  4359          BRB     50$                             ; Complete the request
                                1826  4360 ;
                                1826  4361 ; Global request
                                1826  4362 ;
                                1826  4363 ; Only a read GSS or read global counters can be on INFOUT queue.
                                1826  4364 ;
                52   00AC D5   0F  1826  4365 40$:     REMQUE  @UCB$Q_XD_INFOUT(R5),R2         ; Get next global request
                          F1   1D  182B  4366          BVS     20$                             ; Br if none waiting
                        0C00   30  182D  4367          BSBW    SET_OUTTIM                      ; Start output wait timer
                  51    2C A2   D0  1830  4368          MOVL    IRP$L_SVAPTE(R2),R1             ; Get system buffer address
                        38 A2   D5  1834  4369          TSTL    IRP$L_MEDIA(R2)                 ; Is this a counters request?
                           D3   13  1837  4370          BEQL    10$                             ; Br if no - read of GSS
                                1839  4371 ;
                                1839  4372 ; Global counters request
                                1839  4373 ;
                50   E90C CF   9E  1839  4374          MOVAB   LINE_COUNTER-<2*5>,R0           ; Get address of counter types
                                183E  4375
                                183E  4376 ;
                                183E  4377 ; Handle DMP error counter types.
                                183E  4378 ;
      53   38 A2   08   00   EA  183E  4379 50$:     FFS     #0,#8,IRP$L_MEDIA(R2),R3        ; Get counter index
                                1844  4380          CLRBIT  R3,IRP$L_MEDIA(R2)              ; Indicate counter complete
                     51   61   D0  1849  4381          MOVL    (R1),R1                         ; Get next data pointer
                          52   DD  184C  4382          PUSHL   R2                              ; Save IRP address
                     81   8043   B0  184E  4383          MOVW    (R0)+[R3],(R1)+                 ; Set next counter type
      52   FE A1   03   0C   EF  1852  4384          EXTZV   #NMA$V_CNT_MAP,#3,-2(R1),R2     ; Get width + bit map
                                1858  4385          CASE    R2,TYPE=B,LIMIT=#2,<-           ; Handle only DMP counter types
                                1858  4386                  60$,-                           ;   8 bit counter (2 per read)
                                1858  4387                  70$,-                           ;   8 bit counter, bit map
                                1858  4388                  80$>                            ;   16 bit counter
                                1862  4389
                     81   54   90  1862  4390 60$:     MOVB    R4,(R1)+                        ; Store 1st 8 bit counter
                     81   6043   B0  1865  4391          MOVW    (R0)[R3],(R1)+                  ; Set next counter type
            54   54   F8 8F   78  1869  4392          ASHL    #-8,R4,R4                       ; Shift down next counter
                     81   54   90  186E  4393          MOVB    R4,(R1)+                        ; Store 2nd 8 bit counter
                          10   11  1871  4394          BRB     90$                             ; Check if all done
                                1873  4395
      52   54   08   08   EF  1873  4396 70$:     EXTZV   #8,#8,R4,R2                     ; Get bit map
                     81   52   9B  1878  4397          MOVZBW  R2,(R1)+                        ; Store bit map
                     81   54   90  187B  4398          MOVB    R4,(R1)+                        ; Store counter value
                          03   11  187E  4399          BRB     90$                             ; Check if all done
                                1880  4400
                     81   54   B0  1880  4401 80$:     MOVW    R4,(R1)+                        ; Store 16 bit counter
                                1883  4402
                          52 8ED0  1883  4403 90$:     POPL    R2                              ; Restore IRP address
                53   2C A2   D0  1886  4404          MOVL    IRP$L_SVAPTE(R2),R3             ; Get system PTE address
                   63   51   D0  188A  4405          MOVL    R1,(R3)                         ; Reset next data pointer
                     38 A2   95  188D  4406          TSTB    IRP$L_MEDIA(R2)                 ; All done?
                          07   12  1890  4407          BNEQ    100$                            ; Br if not
                63   0C A3   9E  1892  4408          MOVAB   P2B_T_DATA(R3),(R3)             ; Else, reset start of data
                        FF77   31  1896  4409          BRW     15$                             ; And complete request
```

```
                        1899  4410
0094 C5   62    0E      1899  4411 100$:    INSQUE  (R2),UCB$Q_XD_INPUTQ(R5)        ; Insert IRP on input Q
          F72F  31      189E  4412          BRW     LOAD_PORT                       ; Request input
```

```
                                18A1  4414          .SBTTL  SCHED_FORK,  Schedule the fork process
                                18A1  4415          .SBTTL  SCHED_FORKC, Schedule the fork process with R3 clear
                                18A1  4416
                                18A1  4417 ;++
                                18A1  4418 ; SCHED_FORK - Schedule the fork process
                                18A1  4419 ; SCHED_FORKC - Schedule the fork process with R3 clear
                                18A1  4420 ;
                                18A1  4421 ; Functional description:
                                18A1  4422 ;
                                18A1  4423 ; This routine is called to schedule the error and I/O completion fork process.
                                18A1  4424 ; The last controller port and CSR values are saved for examination.  If the
                                18A1  4425 ; fork process is already pending, only the last CSR values are saved if there
                                18A1  4426 ; was an error.
                                18A1  4427 ;
                                18A1  4428 ; Inputs:
                                18A1  4429 ;
                                18A1  4430 ;       R3 = Last SEL0 and SEL2 values
                                18A1  4431 ;       R4 = Last SEL4 and SEL6 values
                                18A1  4432 ;       R5 = UCB address
                                18A1  4433 ;
                                18A1  4434 ;       IPL = DIPL or higher
                                18A1  4435 ;
                                18A1  4436 ; Outputs:
                                18A1  4437 ;
                                18A1  4438 ;       R3 is cleared if SCHED_FORKC entry.
                                18A1  4439 ;       R5 = UCB address
                                18A1  4440 ;
                                18A1  4441 ; If XD_BSEL2_V_ERR is set in BSEL2 the following is returned:
                                18A1  4442 ;
                                18A1  4443 ;       UCB$L_XD_SEL0(R5) = SEL0 and SEL2 values
                                18A1  4444 ;       UCB$L_XD_SEL4(R5) = SEL4 and SEL6 values
                                18A1  4445 ;
                                18A1  4446 ;--
                                18A1  4447
                                18A1  4448 SCHED_FORKC:                                 ; Schedule fork process, clr R3
                     53    D4   18A1  4449          CLRL    R3                           ; No device error
                                18A3  4450 SCHED_FORK:                                  ; Schedule fork process
          18 68 A5    03    E2   18A3  4451          BBSS    #UCB$V_XD_FORK_PEND,UCB$W_DEVSTS(R5),10$ ; Br if fork pending
                     55    DD   18A8  4452          PUSHL   R5                           ; Save R5
                     04    10   18AA  4453          BSBB    5$                           ; Setup fork process
                     55  8ED0   18AC  4454          POPL    R5                           ; Restore R5
                         05     18AF  4455          RSB                                  ; Return to caller
                                18B0  4456
     55    000002B6 8F    C0   18B0  4457 5$:       ADDL    #UCB$B_XD_FKB,R5             ; Point to fork block in UCB
                CC'AF    9F   18B7  4458          PUSHAB  B^FORK_PROC                  ; Set address of fork process
           00000000'GF    17   18BA  4459          JMP     G^EXE$FORK                   ; Schedule the fork and return
                                18C0  4460
            05 53    15    E1   18C0  4461 10$:     BBC     #XD_BSEL2_V_ERR+16,R3,20$    ; Br if not error
          02C6 C5    53    7D   18C4  4462          MOVQ    R3,UCB$L_XD_SEL0(R5)         ; Save last CSR values
                         05     18C9  4463 20$:     RSB                                  ; Return to caller
```

```
                                        .SBTTL  FORK_PROC,  Error and completion fork process handling

                                ;++
                                ; FORK_PROC - Error and completion fork processing
                                ;
                                ; Functional description:
                                ;
                                ; This routine is called as a fork process to handle errors and all completions
                                ; pending.
                                ;
                                ; Inputs:
                                ;
                                ;       R3 = Last SEL0 and SEL2 values
                                ;       R4 = Last SEL4 and SEL6 values
                                ;       R5 = UCB address
                                ;
                                ;       IPL = FIPL
                                ;
                                ; Outputs:
                                ;
                                ;       R5 is preserved.
                                ;
                                ;--
                    0BBA'       .WORD   TIMEOUT-.                               ; Offset to timeout routine
                                .ENABL  LSB
                                FORK_PROC:                                      ; Error/completion fork process
55  000002B6 8F    C2          SUBL    #UCB$B_XD_FKB,R5                        ; Point to UCB
                                CLRBIT  #UCB$V_XD_FORK_PEND,UCB$W_DEVSTS(R5) ; Clear fork process flag
     0C 53   15     E1          BBC     #XD_BSEL2_V_ERR+16,R3,5$                ; Br if not an error
                                SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Indicate fatal error
        0082 C5     B6          INCW    UCB$W_ERRCNT(R5)                        ; Bump error counter
           0454     31          BRW     SHUTDOWN_DEV                            ; Shutdown the device
                                ;
                                ; Complete any requests on completion queue
                                ;
          59        DD          5$:     PUSHL   R9                              ; Save R9
52  00B4 D5         0F          10$:    REMQUE  @UCB$Q_XD_POST(R5),R2           ; Get next completed block
             0A     1C                  BVC     20$                             ; Br if got one
           0326     30                  BSBW    HALT_TRIBS                      ; Halt all tribs with errors
           0342     30                  BSBW    START_XMITS                     ; Start xmits for tribs in run
             59 8ED0                    POPL    R9                              ; Restore R9
                05                      RSB                                     ; Return to caller
                                ;
                                ; Found a completed block - finish it
                                ;
17      0A A2       91          20$:    CMPB    IRP$B_TYPE(R2),S^#DYN$C_NET     ; Was it a receive?
             0A     13                  BEQL    30$                             ; Br if yes - complete it
0A      0A A2       91                  CMPB    IRP$B_TYPE(R2),S^#DYN$C_IRP     ; Was it an IRP?
             4F     13                  BEQL    50$                             ; Br if yes - complete it
                                        BUG_CHECK NOBUFPCKT,FATAL               ; Else, fatal error
                                ;
                                ; Receive complete - if there is a pending receive I/O request, complete it.
                                ; Otherwise, queue the buffer and, if enabled, deliver attention AST.
                                ;
50      0C A2       9A          30$:    MOVZBL  RCV_L_BACC(R2),R0               ; Get trib address
             11     12                  BNEQ    32$                             ; Br if no error
           42 A5    A0                  ADDW    UCB$W_DEVBUFSIZ(R5),-           ; Adjust receive buffer quota
```

```
                0136 C5          1914  4522                    UCB$W_XD_QUOTA(R5)                      ;
                50   52    D0    1917  4523            MOVL    R2,R0                                   ; Get address of buffer
00000000'GF     50    16    191A  4524            JSB     G^COM$DRVDEALMEM                        ; Deallocate buffer
                      C8    11    1920  4525            BRB     10$                                     ; Get next completion
                                  1922  4526
                    059F    30    1922  4527 32$:     BSBW    FIND_TRIB                               ; Get CDB address
                  04 50     E8    1925  4528            BLBS    R0,33$                                  ; Br if got CDB
                                  1928  4529            BUG_CHECK NOBUFPCKT,FATAL                       ; Else, fatal error
                                  192C  4530
51    0E A2     3C          192C  4531 33$:     MOVZWL  RCV_L_BACC+2(R2),R1                     ; Get the byte count
38 A9   51      C0          1930  4532            ADDL    R1,CDB_L_BRC(R9)                        ; Update byte count
                      04    1E    1934  4533            BCC     35$                                     ; Br if no overflow
38 A9   01      CE          1936  4534            MNEGL   #1,CDB_L_BRC(R9)                        ; Else, latch it
                              193A  4535 35$:     INCC    CDB_L_DMR(R9)                           ; Update message count
53    18 B9     0F          1942  4536            REMQUE  @CDB_Q_RCV_REQ(R9),R3                   ; Remove waiting IRP
                      05    1D    1946  4537            BVS     40$                                     ; Br if none - queue for later
                    0248    30    1948  4538            BSBW    FINISH_RCV_IO                           ; Else, finish the I/O
                      9D    11    194B  4539            BRB     10$                                     ; Look for next completion
                              194D  4540
24 B9   62      0E          194D  4541 40$:     INSQUE  (R2),@CDB_Q_RCV_MSG+4(R9)               ; Queue receive msg for later
                  091C    30    1951  4542            BSBW    POKE_USER                               ; Deliver ASTs
                      94    11    1954  4543            BRB     10$                                     ; Look for more completions
                              1956  4544 ;
                              1956  4545 ; IRP packet completion - branch to proper completion handling routine.
                              1956  4546 ;
53    52        D0          1956  4547 50$:     MOVL    R2,R3                                   ; Copy IRP address
52    21 A3     9A          1959  4548            MOVZBL  IRP$B_XDFUNC(R3),R2                     ; Get function type
                              195D  4549            $DISPATCH       R2,TYPE=B,-                     ; Dispatch on request type
                              195D  4550            <-
                              195D  4551                    <XD_FC_V_INITT  FORK_PKT>,-             ; Start tributary
                              195D  4552                    <XD_FC_V_INITC  FORK_INITC>,-           ; Start DMP11 controller
                              195D  4553                    <XD_FC_V_STOPT  FORK_STOPT>,-           ; Stop tributary
                              195D  4554                    <XD_FC_V_STOPC  FORK_STOPC>,-           ; Stop DMP11 controller
                              195D  4555                    <XD_FC_V_CHGC   FORK_INITC>,-           ; Change controller parameters
                              195D  4556                    <XD_FC_V_CHGT   FORK_PKT>,-             ; Change tributary parameters
                              195D  4557                    <XD_FC_V_RDMODEM FORK_SSLOT>,-          ; Read modem register
                              195D  4558                    <XD_FC_V_RDTERR FORK_COUNTER>,-         ; Read tributary error counts
                              195D  4559                    <XD_FC_V_RDGERR FORK_COUNTER>,-         ; Read global error counts
                              195D  4560                    <XD_FC_V_RCTERR FORK_COUNTER>,-         ; Read and clear tributary ctrs
                              195D  4561                    <XD_FC_V_RCGERR FORK_COUNTER>,-         ; Read and clear global counts
                              195D  4562                    <XD_FC_V_RDTSS  FORK_SSLOT>,-           ; Read TSS location
                              195D  4563                    <XD_FC_V_RDGSS  FORK_SSLOT>,-           ; Read GSS location
                              195D  4564                    <XD_FC_V_WTMODEM FORK_PKT>,-            ; Write modem request
                              195D  4565                    <XD_FC_V_HALTT  FORK_HALTT>,-           ; Halt tributary
                              195D  4566                    <XD_FC_V_KILLT  FORK_KILLT>,-           ; Kill tributary
                              195D  4567                    <XD_FC_V_CANCEL FORK_CANCEL>,-          ; CANCEL request to halt trib
                              195D  4568                    <XD_FC_V_XMIT   FORK_XMIT>,-            ; Xmit request
                              195D  4569                    <XD_FC_V_RECV   FORK_RECV>,-            ; Receive request
                              195D  4570                    <XD_FC_V_NUMSYNC FORK_INITC>,-          ; Set number of sync characters
                              195D  4571                    <XD_FC_V_POLSTAT FORK_PKT>,-            ; Latch polling state
                              195D  4572                    <XD_FC_V_POLSS  FORK_POLSS>,-           ; Read polling sub-state
                              195D  4573            >
                              198D  4574 ;
                              198D  4575 ; Otherwise if not a valid IRP
                              198D  4576 ;
                              198D  4577            BUG_CHECK NOBUFPCKT,FATAL                       ; Fatal driver error
                              1991  4578
```

```
                        1991  4579 FORK_PKT:                                         ; Process completion on packet
     50   42 A3   3C    1991  4580          MOVZWL  IRP$W_CSTATUS(R3),R0             ; Get status of completion
           0238   30    1995  4581          BSBW    IO_DONE                          ; Complete the I/O
           FF4F   31    1998  4582          BRW     10$                              ; Get next packet
                        1998  4583
                        1998  4584 FORK_INITC:                                       ; Process ctrl init completion
     39 A3   01   91    1998  4585          CMPB    S^#XD_FC_V_INITC,IRP$L_MEDIA+1(R3) ; Was this an init controller?
                F0   12  199F  4586          BNEQ    FORK_PKT                         ; Br if not
           42 A3   B5    19A1  4587          TSTW    IRP$W_QUOTA(R3)                  ; Any quota to return?
                27   13  19A4  4588          BEQL    55$                              ; Br if no
     50   0C A3   3C    19A6  4589          MOVZWL  IRP$L_PID(R3),R0                 ; Get process index from IRP
51   00000000'GF   D0    19AA  4590          MOVL    G^SCH$GL_PCBVEC,R1               ; Get address of PCB addr vec
     50   6140   D0    19B1  4591          MOVL    (R1)[R0],R0                      ; Get PCB address
           60 A0   D1    19B5  4592          CMPL    PCB$L_PID(R0),-                  ; Still same process?
                0C A3          19B8  4593                  IRP$L_PID(R3)            ;
                11   12  19BA  4594          BNEQ    55$                              ; Br if not - forget it
     50   0080 C0   D0    19BC  4595          MOVL    PCB$L_JIB(R0),R0                 ; Get JIB address
     51   42 A3   3C    19C1  4596          MOVZWL  IRP$W_QUOTA(R3),R1               ; Convert quota to longword
     20 A0   51   C0    19C5  4597          ADDL    R1,JIB$L_BYTCNT(R0)              ; Return byte count quota
     24 A0   51   C0    19C9  4598          ADDL    R1,JIB$L_BYTLM(R0)               ; ..and byte limit quota
                        19CD  4599
     50   42 A3   3C    19CD  4600 55$:     MOVZWL  IRP$W_CSTATUS(R3),R0             ; Get completion status
                50   DD  19D1  4601          PUSHL   R0                               ; Save completion status
           01FA   30    19D3  4602          BSBW    IO_DONE                          ; Complete the I/O request
                50 8ED0   19D6  4603          POPL    R0                               ; Restore status
           03 50   E9    19D9  4604          BLBC    R0,57$                           ; Br ir error on startup
            F4FA   30    19DC  4605          BSBW    FILLRCVLIST                      ; Start the receives on success
            FF08   31    19DF  4606 57$:     BRW     10$                              ; Get next packet
                        19E2  4607
                        19E2  4608 FORK_STOPT:                                       ; Process stop of tributary
             056B   30    19E2  4609          BSBW    GET_CDB                          ; Get CDB address
           24 50   E9    19E5  4610          BLBC    R0,58$                           ; Br if no CDB
             026D   30    19E8  4611          BSBW    SHUTDOWN_TRIB1                   ; Abort all I/O
          50   01   3C    19EB  4612          MOVZWL  S^#SS$_NORMAL,R0                 ; Set completion status
                        19EE  4613 ;
                        19EE  4614 ; Cleanup CDB in case we use it again
                        19EE  4615 ;
             0C A9   B4    19EE  4616          CLRW    CDB_W_STS(R9)                    ; Clean up status word
69   18   08   00   F0    19F1  4617          INSV    #0,#8,#24,CDB_L_DEVDEPEND(R9)    ; Clean up device status
           38 A9   7C    19F6  4618          CLRQ    CDB_L_CNTS(R9)                   ; Clean up counts
           40 A9   7C    19F9  4619          CLRQ    CDB_L_CNTS+8(R9)                 ;  ...
  0F A9   0135 C5   90    19FC  4620          MOVB    UCB$B_XD_XMT_TRB(R5),CDB_B_XMT_CNT(R9) ; Init max xmits allowed
    05 44 A5   06   E0    1A02  4621          BBS     #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5),58$ ; Br if control station
    0F A9   7F 8F   90    1A07  4622          MOVB    #INF_XMT,CDB_B_XMT_CNT(R9)       ; Else, allow infinite xmits
             01C1   30    1A0C  4623 58$:     BSBW    IO_DONE                          ; Complete the I/O request
             FED8   31    1A0F  4624          BRW     10$                              ; Get next packet
                        1A12  4625
                        1A12  4626 FORK_STOPC:                                       ; Process stop of controller
             0327   30    1A12  4627          BSBW    SHUTDOWN_DEV                     ; Abort all I/O on all tribs
          50   01   3C    1A15  4628          MOVZWL  S^#SS$_NORMAL,R0                 ; Set completion status
             01B5   30    1A18  4629          BSBW    IO_DONE                          ; Complete the I/O request
             FECC   31    1A1B  4630          BRW     10$                              ; Get next packet
                        1A1E  4631
                        1A1E  4632 FORK_COUNTER:                                     ; Process counter done
     50   32 A3   3C    1A1E  4633          MOVZWL  IRP$W_BCNT(R3),R0                ; Get size of P2 buffer
           32 A3   B4    1A22  4634          CLRW    IRP$W_BCNT(R3)                   ; Assume error - no data return
        18 42 A3   E9    1A25  4635          BLBC    IRP$W_CSTATUS(R3),70$            ; Br if err when getting counts
```

```
                32 A3    50   B0  1A29  4636          MOVW    R0,IRP$W_BCNT(R3)               ; Else, reset size of data
                50   3A A3   B1  1A2D  4637          CMPW    IRP$L_MEDIA+2(R3),R0            ; Is user buffer larger than P2?
                       0A    1E  1A31  4638          BGEQU   60$                             ; Br if yes - okay
                50   3A A3   B0  1A33  4639          MOVW    IRP$L_MEDIA+2(R3),R0            ; Else, set user return size
             42 A3    0601 8F   B0  1A37  4640          MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3) ; Set error return
             50   50     10   78  1A3D  4641 60$:      ASHL    #16,R0,R0                       ; Move count to high word
                50   42 A3   B0  1A41  4642 70$:      MOVW    IRP$W_CSTATUS(R3),R0            ; Set return status
                      0188   30  1A45  4643          BSBW    IO_DONE                         ; Complete the I/O request
                       FE9F  31  1A48  4644          BRW     10$                             ; Get next packet
                             1A48  4645
                             1A48  4646 FORK_SSLOT:                                         ; Process status slot read
                50   42 A3   3C  1A48  4647          MOVZWL  IRP$W_CSTATUS(R3),R0            ; Get completion status
                      03 50    E8  1A4F  4648          BLBS    R0,75$                          ; Br if success
                      32 A3    94  1A52  4649          CLRB    IRP$W_BCNT(R3)                  ; Else, no data to return
                       0178   30  1A55  4650 75$:      BSBW    IO_DONE                         ; Complete the I/O request
                       FE8F   31  1A58  4651          BRW     10$                             ; Get next packet
                             1A5B  4652
                             1A5B  4653 FORK_XMIT:                                          ; Process transmit done
                       04F2   30  1A5B  4654          BSBW    GET_CDB                         ; Get CDB address
                      06 50    E8  1A5E  4655          BLBS    R0,78$                          ; Br if CDB present
                42 A3    50   B0  1A61  4656          MOVW    R0,IRP$W_CSTATUS(R3)            ; Else, set error return
                         1D   11  1A65  4657          BRB     90$                             ; Return map registers
                             1A67  4658
                   0F A9     96  1A67  4659 78$:      INCB    CDB_B_XMT_CNT(R9)               ; Indicate another xmit complete
                16 42 A3     E9  1A6A  4660          BLBC    IRP$W_CSTATUS(R3),90$           ; Br if xmit error
             51    32 A3     3C  1A6E  4661          MOVZWL  IRP$W_BCNT(R3),R1               ; Get size of transmit
             3C A9    51     C0  1A72  4662          ADDL    R1,CDB_L_BSN(R9)                ; Update byte count
                      04     1E  1A76  4663          BCC     80$                             ; Br if no overflow
             3C A9    01     CE  1A78  4664          MNEGL   #1,CDB_L_BSN(R9)                ; Else, latch it
                             1A7C  4665 80$:      INCC    CDB_L_DMS(R9)                   ; Update message count
             51    3C A3     9A  1A84  4666 90$:      MOVZBL  IRP$L_MEDIA+4(R3),R1            ; Get mapping slot number used
                             1A88  4667          CLRBIT  R1,UCB$B_XD_XMT_MAP(R5)         ; Clear in use flag
                             1A8E  4668 ;
                             1A8E  4669 ; For all except uVAX I, we must release the UBA map registers.
                             1A8E  4670 ;
                             1A8E  4671          CPUDISP <<790,93$>,-
                             1A8E  4672                   <780,93$>,-
                             1A8E  4673                   <750,93$>,-
                             1A8E  4674                   <730,93$>,-
                             1A8E  4675                   <UV1,96$>>
                             1AA8  4676
                52    24 A5    D0  1AA8  4677 93$:      MOVL    UCB$L_CRB(R5),R2                ; Get CRB address
          34 A2    00E0 C541   D0  1AAC  4678          MOVL    UCB$L_XD_XMT_MAP(R5)[R1],-      ; Setup map register data
                             1AB3  4679                   CRB$L_INTD+VEC$W_MAPREG(R2)     ;
             00E0 C541    01   CE  1AB3  4680          MNEGL   #1,UCB$L_XD_XMT_MAP(R5)[R1]     ; Set mapping data not allocated
                             1AB9  4681          RELMPR                                  ; Release the map registers
                             1ABF  4682
                             1ABF  4683 96$:
                50   42 A3    3C  1ABF  4684          MOVZWL  IRP$W_CSTATUS(R3),R0            ; Get return status
                      0B 50    E9  1AC3  4685          BLBC    R0,100$                         ; Br if error
                50   32 A3    B0  1AC6  4686          MOVW    IRP$W_BCNT(R3),R0               ; Get count of bytes transmitted
             50    50    10   78  1ACA  4687          ASHL    #16,R0,R0                       ; Shift into place
                50    01     B0  1ACE  4688          MOVW    S^#SS$_NORMAL,R0                ; Set success
                   00FC       30  1AD1  4689 100$:     BSBW    IO_DONE                         ; Complete the I/O
                   E824       30  1AD4  4690          BSBW    XMT_ALT_START                   ; Give other xmits to device
                   FE10       31  1AD7  4691          BRW     10$                             ; Get next packet
                             1ADA  4692
```

```
                        1ADA  4693 FORK_RECV:                                    ; Process receive request
             0473   30  1ADA  4694          BSBW    GET_CDB                     ; Get CDB address
           0C 50    E9  1ADD  4695          BLBC    R0,T10$                     ; Br if error
       50  42 A3   3C  1AE0  4696          MOVZWL  IRP$W_CSTATUS(R3),R0        ; Get completion status
           05 50    E9  1AE4  4697          BLBC    R0,110$                     ; Br if error
            E9A2    30  1AE7  4698          BSBW    RCV_START_ALT               ; Get any receives waiting
              03    11  1AEA  4699          BRB     120$                        ; Continue
                        1AEC  4700
            00E1    30  1AEC  4701 110$:    BSBW    IO_DONE                     ; Complete the I/O request
            FDF8    31  1AEF  4702 120$:    BRW     10$                         ; Get next packet
                        1AF2  4703
                        1AF2  4704 FORK_HALTT:
         2E 42 A3   E9  1AF2  4705          BLBC    IRP$W_CSTATUS(R3),FORK_KILLT ; Br if error
       21 A3    0F  90  1AF6  4706          MOVB    #XD_FC_V_KILLT,IRP$B_XDFUNC(R3) ; Set to kill trib
                        1AFA  4707          DSBINT  UCB$B_DIPL(R5)              ; Sync access to device
  0098 D5       63  0E  1B01  4708          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5) ; Insert at end of input Q
            F4C7    30  1B06  4709          BSBW    LOAD_PORT                   ; Give request to device
                        1B09  4710          ENBINT                              ; Restore IPL
            FDDB    31  1B0C  4711          BRW     10$                         ; Get next packet
                        1B0F  4712
                        1B0F  4713 FORK_CANCEL:                                 ; Process completion of $CANCEL
            043E    30  1B0F  4714          BSBW    GET_CDB                     ; Get CDB address
          0F 50     E9  1B12  4715          BLBC    R0,FORK_KILLT               ; Br if error
             0D     E1  1B15  4716          BBC     #XM$V_STS_RUNNING,-         ; Br if not in
          0B 69          1B17  4717                  CDB_L_DEVDEPEND(R9),FORK_KILLT ;   run state
        0228 8F     BB  1B19  4718          PUSHR   #^M<R3,R5,R9>               ; Save registers
            E7DB    30  1B1D  4719          BSBW    XMT_ALT_START               ; Start up any pending xmits
        0228 8F     BA  1B20  4720          POPR    #^M<R3,R5,R9>               ; Restore registers
                        1B24  4721
                        1B24  4722 FORK_KILLT:                                  ; Process completion of kill
                OCDA    30  1B24  4723          BSBW    POST_IT                 ; Deallocate CDB from IOPOST
                FDC0    31  1B27  4724          BRW     10$                     ; Get next packet
                        1B2A  4725
                        1B2A  4726 FORK_POLSS:                                  ; Process Poll substate
             22 42 A3   E9  1B2A  4727          BLBC    IRP$W_CSTATUS(R3),130$      ; Br if error
                   06   E1  1B2E  4728          BBC     #XM$V_CHR_CTRL,-            ; Br if not control station
             1D 44 A5          1B30  4729                  UCB$L_DEVDEPEND(R5),130$ ;   ...
  50   3F A3    08   00  EA  1B33  4730          FFS     #0,#8,IRP$B_POLS(R3),R0     ; Find state
                   15   13  1B39  4731          BEQL    130$                        ; Br if no-state????
                        1B3B  4732          ASSUME  P2B_L_POINTER EQ 0
          51    2C B3   D0  1B3B  4733          MOVL    @IRP$L_SVAPTE(R3),R1        ; Get P2 buffer address
       66 A1    03F3 8F   B0  1B3F  4734          MOVW    #NMA$C_PCCI_PLS,TRIB_PRM_BUFSIZ(R1) ; Stuff parameter code
    68 A1    E5FD CF40   9A  1B45  4735          MOVZBL  POLSS_TBL[R0],TRIB_PRM_BUFSIZ+2(R1) ; Return parameter value
                   02   13  1B4C  4736          BEQL    130$                        ; Br if UNKNOWN
                   04   11  1B4E  4737          BRB     140$                        ; Continue
          32 A3    06   A2  1B50  4738 130$:   SUBW    #6,IRP$W_BCNT(R3)           ; No polling substate return
          50   32 A3   B0  1B54  4739 140$:   MOVW    IRP$W_BCNT(R3),R0           ; Assume user has good size buffer
          50   3C A3   B1  1B58  4740          CMPW    IRP$L_MEDIA+4(R3),R0        ; Is user buffer large enough?
                   0E   1E  1B5C  4741          BGEQU   150$                        ; Br if yes
       42 A3    0601 8F   B0  1B5E  4742          MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3) ; Return partial success
          50    3C A3   B0  1B64  4743          MOVW    IRP$L_MEDIA+4(R3),R0        ; Set size of return
          32 A3    50   B0  1B68  4744          MOVW    R0,IRP$W_BCNT(R3)           ;   ...
        50   50    10   78  1B6C  4745 150$:   ASHL    #16,R0,R0                   ; Shift size of buffer return
          50   42 A3   B0  1B70  4746          MOVW    IRP$W_CSTATUS(R3),R0        ; Set success status
                   50   DD  1B74  4747          PUSHL   R0                          ; Save return status
          52   38 A3   D0  1B76  4748          MOVL    IRP$L_MEDIA(R3),R2          ; Retrieve P1 buffer address
                   0E   13  1B7A  4749          BEQL    170$                        ; Br if none
```

```
                62    40 A5    7D  1B7C  4750          MOVQ    UCB$B_DEVCLASS(R5),(R2)         ; Else, return characteristics
                      03CD    30  1B80  4751          BSBW    GET_CDB                         ; Get CDB address
                   04 50      E9  1B83  4752          BLBC    R0,170$                         ; Br if error
                04 A2   69    C8  1B86  4753          BISL    CDB_L_DEVDEPEND(R9),4(R2)       ;       ...
                      50 8ED0  1B8A  4754 170$:       POPL    R0                              ; Restore return status
                    0040      30  1B8D  4755          BSBW    IO_DONE                         ; Complete the I/O request
                    FD57      31  1B90  4756          BRW     10$                             ; Get next packet
                              1B93  4757          .DSABL  LSB
```

```
                     1B93  4759          .SBTTL  FINISH_RCV_IO,  Finish receive I/O processing
                     1B93  4760
                     1B93  4761 ;++
                     1B93  4762 ; FINISH_RCV_IO - Finish receive I/O processing
                     1B93  4763 ;
                     1B93  4764 ; Functional description:
                     1B93  4765 ;
                     1B93  4766 ; This routine completes a receive operation that has been matched with a
                     1B93  4767 ; message block. After the receive has been completed the message free list
                     1B93  4768 ; is filled and a receive is started if needed.
                     1B93  4769 ;
                     1B93  4770 ;  Inputs:
                     1B93  4771 ;
                     1B93  4772 ;       R2 = message buffer address
                     1B93  4773 ;       R3 = I/O packet address
                     1B93  4774 ;       R5 = UCB address
                     1B93  4775 ;       R9 = CDB address
                     1B93  4776 ;
                     1B93  4777 ;       IPL = FIPL
                     1B93  4778 ;
                     1B93  4779 ;  Outputs:
                     1B93  4780 ;
                     1B93  4781 ;       R5 is preserved.
                     1B93  4782 ;
                     1B93  4783 ;       The request is completed via I/O post.
                     1B93  4784 ;--
                     1B93  4785
                     1B93  4786 FINISH_RCV_IO:                                  ; Finish recieve I/O request
    2C A3    52  D0  1B93  4787          MOVL    R2,IRP$L_SVAPTE(R3)           ; Save block address
    62   48 A2  9E  1B97  4788          MOVAB   RCV_T_DATA(R2),(R2)           ; Set address of received data
04 A2    38 A3  D0  1B9B  4789          MOVL    IRP$L_MEDIA(R3),4(R2)         ; Set address of user buffer
      42 A5  A0  1BA0  4790          ADDW    UCB$W_DEVBUFSIZ(R5),-         ; Adjust receive buffer quota
          0136 C5       1BA3  4791                  UCB$W_XD_QUOTA(R5)            ;
   51    0E A2  B0  1BA6  4792          MOVW    RCV_L_BACC+2(R2),R1           ; Get size of transfer
   32 A3    51  B1  1BAA  4793          CMPW    R1,IRP$W_BCNT(R3)             ; Request larger than actual?
            04  1B  1BAE  4794          BLEQU   10$                           ; Br if no
   51    32 A3  3C  1BB0  4795          MOVZWL  IRP$W_BCNT(R3),R1             ; Set size to minimum of two
   32 A3    51  B0  1BB4  4796 10$:     MOVW    R1,IRP$W_BCNT(R3)             ; Set size to transfer
50   51    10  78  1BB8  4797          ASHL    #16,R1,R0                     ; Set buffer size in status
            07  12  1BBC  4798          BNEQ    20$                           ; Br if success
50   0054 8F  B0  1BBE  4799          MOVW    #SS$_CTRLERR,R0               ; Set data path error
            03  11  1BC3  4800          BRB     30$                           ; Fill receive list
                1BC5  4801
      50    01  B0  1BC5  4802 20$:     MOVW    S^#SS$_NORMAL,R0              ; Set success
            06  10  1BC8  4803 30$:     BSBB    IO_DONE                       ; Post the I/O request
         F30C  31  1BCA  4804          BRW     FILLRCVLIST                   ; Fill up the receive buffers
                1BCD  4805 ;
                1BCD  4806 ; Complete an I/O request packet
                1BCD  4807 ;
                1BCD  4808 ABORT_PKT:                                      ; Abort the I/O request
      50    2C  3C  1BCD  4809          MOVZWL  #SS$_ABORT,R0                 ; Return aborted status
                1BD0  4810 IO_DONE:                                        ; Complete a transfer I/O request
   38 A3    50  D0  1BD0  4811          MOVL    R0,IRP$L_IOST1(R3)            ; Set status return and size
                1BD4  4812 IO_DONE1:                                       ; Alternate entry point
3C A3    44 A5  D0  1BD4  4813          MOVL    UCB$L_DEVDEPEND(R5),IRP$L_IOST2(R3) ; Set other info
   50    44 A3  D0  1BD9  4814          MOVL    IRP$L_CDB(R3),R0              ; Get address of CDB address
            09  13  1BDD  4815          BEQL    10$                           ; Br if none
```

```
                        1BDF  4816
          50   60   D0   1BDF  4817          MOVL    (R0),R0                          ; Get CDB address
               04   13   1BE2  4818          BEQL    10$                              ; Br if none
        3C A3   60   C8   1BE4  4819          BISL    CDB_L_DEVDEPEND(R0),IRP$L_IOST2(R3) ; Set circuit bits
    14 2A A3   07   E1   1BE8  4820 10$:      BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),20$ ; Br if no diagnostic buffer
 50   4C B3   08   C1   1BED  4821          ADDL3   #8,@IRP$L_DIAGBUF(R3),R0         ; Address buffer past start time
80   00000000'GF   7D   1BF2  4822          MOVQ    G^EXE$GQ_SYSTIME,(R0)+           ; Insert stop time
    80   0082 C5   3C   1BF9  4823          MOVZWL  UCB$W_ERRCNT(R5),(R0)+           ; Insert error counter
            0239   30   1BFE  4824          BSBW    REG_DUMP                         ; Dump registers
                   1C01  4825 20$:
                   1C01  4826 POST_IT:                                             ; Post IRP for completion
                   1C01  4828          PUSHQ   R6                                  ; Save registers
          50   05   90   1C04  4829          MOVB    #JNL_POST,R0                     ; Set journal type = POST
             08C3   30   1C07  4830          BSBW    XD_FILL_JNX                      ; Fill the journal buffer
           04 50   E9   1C0A  4831          BLBC    R0,50$                           ; Br if error
        66   38 A3   7D   1C0D  4832          MOVQ    IRP$L_IOST1(R3),(R6)             ; Store the IOSB
                   1C11  4833 50$:     POPQ    R6                                  ; Restore registers
                   1C14  4835
     00000000'GF   17   1C14  4836          JMP     G^COM$POST                       ; Post the I/O and return
```

```
                                1C1A  4838          .SBTTL  HALT_TRIBS,  Halt all tribs that were forced to shutdown
                                1C1A  4839
                                1C1A  4840 ;++
                                1C1A  4841 ; HALT_TRIBS - Halt all tribs that were forced to shutdown
                                1C1A  4842 ;
                                1C1A  4843 ; Functional description:
                                1C1A  4844 ;
                                1C1A  4845 ; This routine scans the UCB halt longword for any trib that were recently
                                1C1A  4846 ; shutdown due to device errors.
                                1C1A  4847 ;
                                1C1A  4848 ; Inputs:
                                1C1A  4849 ;
                                1C1A  4850 ;       R5 = UCB address
                                1C1A  4851 ;
                                1C1A  4852 ;       IPL = FIPL
                                1C1A  4853 ;
                                1C1A  4854 ; Outputs:
                                1C1A  4855 ;
                                1C1A  4856 ;       R0-R2,R9 are destroyed.
                                1C1A  4857 ;
                                1C1A  4858 ;--
                                1C1A  4859
                                1C1A  4860 HALT_TRIBS:                                  ; Shutdown all tribs that are halted
52   029C C5   E532 CF   00    EA  1C1A  4861          FFS     #0,MAX_W_TRIB,UCB$L_XD_HALT(R5),R2 ; Get index of next halted trib
                          13    13  1C23  4862          BEQL    10$                     ; Br if none
                                1C25  4863          CLRBIT  R2,UCB$L_XD_HALT(R5)    ; Else, indicate trib shutdown
          59   0218 C542    D0  1C2B  4864          MOVL    UCB$L_XD_CDB_VEC(R5)[R2],R9 ; Get CDB address
                        E7    13  1C31  4865          BEQL    HALT_TRIBS              ; Br if no CDB address
                      0039    30  1C33  4866          BSBW    SHUTDOWN_TRIB           ; Shutdown the tributary
                        E2    11  1C36  4867          BRB     HALT_TRIBS              ; Look for next halted trib
                              05  1C38  4868 10$:     RSB                             ; Return to caller
```

```
                                1C39  4870          .SBTTL  START_XMITS,  Start xmits for tribs in run state
                                1C39  4871
                                1C39  4872 ;++
                                1C39  4873 ; START_XMITS - Start xmits for tribs in run state
                                1C39  4874 ;
                                1C39  4875 ; Functional description:
                                1C39  4876 ;
                                1C39  4877 ; This routine scans the UCB run longword for any tribs that have recently
                                1C39  4878 ; entered the run state, to start the xmit requests for that tributary.
                                1C39  4879 ;
                                1C39  4880 ;  Inputs:
                                1C39  4881 ;
                                1C39  4882 ;       R5 = UCB address
                                1C39  4883 ;
                                1C39  4884 ;       IPL = FIPL
                                1C39  4885 ;
                                1C39  4886 ;  Outputs:
                                1C39  4887 ;
                                1C39  4888 ;       R0-R2,R9 are destroyed.
                                1C39  4889 ;
                                1C39  4890 ;--
                                1C39  4891
                                1C39  4892 START_XMITS:                                 ; Start xmits on running tribs
52  0298 C5  E513 CF  00  EA    1C39  4893          FFS     #0,MAX_W_TRIB,UCB$L_XD_RUN(R5),R2 ; Get index of next running trib
                        13  13   1C42  4894          BEQL    10$                         ; Br if none running
                                1C44  4895          CLRBIT  R2,UCB$L_XD_RUN(R5)         ; Else, indicate xmits started
            59  0218 C542  D0    1C4A  4896          MOVL    UCB$L_XD_CDB_VEC(R5)[R2],R9 ; Get CDB address
                      E7  13    1C50  4897          BEQL    START_XMITS                 ; Br if no CDB address
                    E6A6  30    1C52  4898          BSBW    XMT_ACT_START               ; Start xmits for this trib
                      E2  11    1C55  4899          BRB     START_XMITS                 ; Get next trib
                          05    1C57  4900 10$:     RSB                                 ; Return to caller
```

```
                        1C58  4902          .SBTTL  SHUTDOWN_TRIB,  Shutdown the tributary
                        1C58  4903
                        1C58  4904 ;++
                        1C58  4905 ; SHUTDOWN_TRIB - Shutdown the tributary
                        1C58  4906 ;
                        1C58  4907 ; Functional description:
                        1C58  4908 ;
                        1C58  4909 ; This routine is called to abort all I/O pending for this tributary.
                        1C58  4910 ;
                        1C58  4911 ; Inputs:
                        1C58  4912 ;
                        1C58  4913 ;       R5 = UCB address
                        1C58  4914 ;       R9 = CDB address
                        1C58  4915 ;
                        1C58  4916 ;       IPL = FIPL
                        1C58  4917 ;
                        1C58  4918 ; Outputs:
                        1C58  4919 ;
                        1C58  4920 ;       R0-R2 are destroyed.
                        1C58  4921 ;
                        1C58  4922 ;--
                        1C58  4923
                        1C58  4924 SHUTDOWN_TRIB1:
                        1C58  4925          PUSHQ   R3                          ; Save R3, R4
      12    10          1C5B  4926          BSBB    SHUTDOWN_TRIB               ; Shutdown the tributary
53  30 B9   0F          1C5D  4927 20$:     REMQUE  @CDB_Q_INFOUT(R9),R3        ; Get any leftover IRPs
      05    1D          1C61  4928          BVS     50$                         ; Br if none
    FF67    30          1C63  4929          BSBW    ABORT_PKT                   ; Else, abort the IRP
      F5    11          1C66  4930          BRB     20$                         ; Look for more
                        1C68  4931 50$:     POPQ    R3                          ; Restore R3, R4
    07C2    30          1C6B  4932          BSBW    SET_OUTTIM                  ; Reset output wait timer
                05      1C6E  4933          RSB
                        1C6F  4934
                        1C6F  4935 SHUTDOWN_TRIB:                               ; Shutdown a tributary
01 0C A9    00      E0  1C6F  4936          BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$ ; Br if trib established
                05      1C74  4937          RSB
                        1C75  4938 ;
                        1C75  4939 ; Flush all attention ASTs
                        1C75  4940 ;
      00D8  8F      BB  1C75  4941 10$:     PUSHR   #^M<R3,R4,R6,R7>            ; Save registers
   57   04  A9      9E  1C79  4942 20$:     MOVAB   CDB_L_AST(R9) R7            ; Get address of AST listhead
   50   67          D0  1C7D  4943          MOVL    (R7),R0                     ; Anything in list?
        1B          13  1C80  4944          BEQL    30$                         ; Br if not
   56   22  A0      3C  1C82  4945          MOVZWL  ACB$L_KAST+10(R0),R6        ; Force channel match
   52   24  A0      3C  1C86  4946          MOVZWL  ACB$L_KAST+12(R0),R2        ; Get process index
54 00000000'GF      D0  1C8A  4947          MOVL    G^SCH$GL_PCBVEC,R4          ; Get PCB address vector address
   54   6442        D0  1C91  4948          MOVL    (R4)[R2],R4                 ; Get PCB address
00000000'GF        16  1C95  4949          JSB     G^COM$FLUSHATTNS            ; Flush AST
           DC       11  1C9B  4950          BRB     20$                         ; Continue flushing ASTs
                        1C9D  4951 ;
                        1C9D  4952 ; Complete all IRPs for this tributary
                        1C9D  4953 ;
00002800  8F        CA  1C9D  4954 30$:     BICL    #XM$M_STS_ACTIVE!XM$M_STS_RUNNING,- ; Clear the running
                65      1CA3  4955                  CDB_L_DEVDEPEND(R5)         ;  and active flags
   56   10  A9      9E  1CA4  4956          MOVAB   CDB_Q_QUEUES(R9),R6         ; Get address of first queue listhea
                        1CA8  4957          ASSUME  CDB_Q_INFOUT EQ CDB_Q_QUEUES+<8*<CDB_C_QUEUES-1>>
                        1CA8  4958          ASSUME  CDB_Q_XMT_PND EQ CDB_Q_QUEUES+<8*<CDB_C_QUEUES-2>>
```

```
                57   03    3C  1CA8  4959          MOVZWL  #CDB_C_QUEUES-2,R7                ; Get number of queues
            53  00 B6    0F  1CAB  4960 40$:      REMQUE  @(R6),R3                        ; Get next IRP/buffer
                   23    1D  1CAF  4961          BVS     80$                             ; Br if none - queue empty
            0A  0A A3    91  1CB1  4962          CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP     ; Is this an IRP?
                   0A    13  1CB5  4963          BEQL    50$                             ; Br if yes
            17  0A A3    91  1CB7  4964          CMPB    IRP$B_TYPE(R3),S^#DYN$C_NET     ; Is it a receive buffer?
                   09    13  1CBB  4965          BEQL    70$                             ; Br if yes
                         1CBD  4966          BUG_CHECK NOBUFPCKT,FATAL                   ; Else, fatal error
                         1CC1  4967 ;
                         1CC1  4968 ; IRP
                         1CC1  4969 ;
                     FF09  30    1CC1  4970 50$:      BSBW    ABORT_PKT                   ; Abort the I/O request
                       E5  11    1CC4  4971          BRB     40$                             ; Get next entry
                         1CC6  4972 ;
                         1CC6  4973 ; Receive buffer
                         1CC6  4974 ;
                52   53    D0  1CC6  4975 70$:      MOVL    R3,R2                           ; Set receive buffer address
                   42 A5    A0  1CC9  4976          ADDW    UCB$W_DEVBUFSIZ(R5),-           ; Adjust receive buffer quota
                0136 C5         1CCC  4977                  UCB$W_XD_QUOTA(R5)              ;
                   F20E  30    1CCF  4978          BSBW    ADDRCVLIST                      ; Try to add to receive list
                     D7  11    1CD2  4979          BRB     40$                             ; Get next entry
                         1CD4  4980 ;
                         1CD4  4981 ; Loop to next queue
                         1CD4  4982 ;
                56   08    C0  1CD4  4983 80$:      ADDL    #8,R6                           ; Skip to next queue listhead
                  D1 57    F5  1CD7  4984          SOBGTR  R7,40$                          ; Loop if more queues
                         1CDA  4985 ;
                         1CDA  4986 ; Abort the transmit requests in progress and return the map registers
                         1CDA  4987 ; and transmit slots!
                         1CDA  4988 ;
                53   28 B9    0F  1CDA  4989 90$:      REMQUE  @CDB_Q_XMT_PND(R9),R3           ; Get next xmit request
                        26    1D  1CDE  4990          BVS     100$                            ; Br if none
                51   3C A3    9A  1CE0  4991          MOVZBL  IRP$L_MEDIA+4(R3),R1            ; Get mapping slot number used
                         1CE4  4992          CLRBIT  R1,UCB$B_XD_XMT_MAP(R5)         ; Clear in use flag
                52   24 A5    D0  1CEA  4993          MOVL    UCB$L_CRB(R5),R2                ; Get CRB address
            34 A2   00E0 C541    D0  1CEE  4994          MOVL    UCB$L_XD_XMT_MAP(R5)[R1],-      ; Setup map register data
                         1CF5  4995                  CRB$L_INTD+VEC$W_MAPREG(R2)
               00E0 C541    01  CE  1CF5  4996          MNEGL   #1,UCB$L_XD_XMT_MAP(R5)[R1]     ; Set mapping data not allocated
                         1CFB  4997          RELMPR                                  ; Release the map registers
                     FEC9  30    1D01  4998          BSBW    ABORT_PKT                       ; Abort the I/O request
                       D4  11    1D04  4999          BRB     90$                             ; Loop til done
                         1D06  5000 100$:
                         1D06  5001 ;
                         1D06  5002 ; Take special care not to abort any shutdown requests that are pending
                         1D06  5003 ;
                56   30 A9    9E  1D06  5004          MOVAB   CDB_Q_INFOUT(R9),R6             ; Get address of infout queue
                   53   66    D0  1D0A  5005 140$:      MOVL    (R6),R3                         ; Get address of next in queue
                   56   53    D1  1D0D  5006 150$:      CMPL    R3,R6                           ; At end of queue?
                        1D    13  1D10  5007          BEQL    180$                            ; Br if yes, all done
                21 A3   02    91  1D12  5008          CMPB    #XD_FC_V_STOPT,IRP$B_XDFUNC(R3) ; Is this a STOPT request?
                        12    13  1D16  5009          BEQL    170$                            ; Br if yes
                21 A3   10    91  1D18  5010          CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R3) ; Is this a CANCEL request?
                        04    12  1D1C  5011          BNEQ    160$                            ; Br if no
                0C A9   04    AA  1D1E  5012          BICW    #CD_TS_M_CANCEL,CDB_W_STS(R9)   ; Indicate that CANCEL is done
                   53   63    0F  1D22  5013 160$:      REMQUE  (R3),R3                         ; Else, remove from queue
                     FEA5  30    1D25  5014          BSBW    ABORT_PKT                       ; And abort the I/O request
                       E0  11    1D28  5015          BRB     140$                            ; Look for more
```

```
                53   63    D0  1D2A  5016 170$:   MOVL    (R3),R3                         ; Move down list
                     DE    11  1D2D  5017         BRB     150$                            ; Continue
                                1D2F  5018 ;
                                1D2F  5019 ; Restart output timers if not a fatal device error
                                1D2F  5020 ;
          03 44 A5   10    E0  1D2F  5021 180$:   BBS     #XMSV_ERR_FATAL,UCB$L_DEVDEPEND(R5),190$ ; Br if fatal error
                   06F9    30  1D34  5022         BSBW    SET_OUTTIM                      ; Reset output timer
                00D8 8F    BA  1D37  5023 190$:   POPR    #^M<R3,R4,R6,R7>                ; Restore registers
                           05  1D3B  5024         RSB                                     ; Return to caller
```

```
                                1D3C  5026          .SBTTL  SHUTDOWN_DEV,  Shutdown the controller
                                1D3C  5027
                                1D3C  5028 ;++
                                1D3C  5029 ; SHUTDOWN_DEV - Shutdown the controller
                                1D3C  5030 ;
                                1D3C  5031 ; Functional description:
                                1D3C  5032 ;
                                1D3C  5033 ; This routine is called to abort all I/O pending on all tributaries.
                                1D3C  5034 ;
                                1D3C  5035 ; Inputs:
                                1D3C  5036 ;
                                1D3C  5037 ;       R5 = UCB address
                                1D3C  5038 ;
                                1D3C  5039 ;       IPL = FIPL
                                1D3C  5040 ;
                                1D3C  5041 ; Outputs:
                                1D3C  5042 ;
                                1D3C  5043 ;       R0-R2 are destroyed.
                                1D3C  5044 ;
                                1D3C  5045 ;--
                                1D3C  5046
                                1D3C  5047 SHUTDOWN_DEV:                                ; Shutdown controller
0A 64 A5    04    E1            1D3C  5048          BBC     #UCB$V_ONLINE,UCB$W_STS(R5),10$ ; Br if not online
06 68 A5    00    E0            1D41  5049          BBS     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),20$ ; Br if device inited
01 68 A5    01    E0            1D46  5050          BBS     #UCB$V_XD_INITING,UCB$W_DEVSTS(R5),20$ ; Br if device initing
                   05           1D48  5051 10$:     RSB
                                1D4C  5052 ;
                                1D4C  5053 ; Abort all I/O for all tribs and deallocate CDBs
                                1D4C  5054 ;
      02D8 8F      BB            1D4C  5055 20$:     PUSHR   #^M<R3,R4,R6,R7,R9>             ; Save registers
   54   24 A5      D0            1D50  5056          MOVL    UCB$L_CRB(R5),R4                ; Get CRB address
                                1D54  5057          ASSUME  IDB$L_CSR EQ 0
   54   2C B4      D0            1D54  5058          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4    ; Get CSR address
01 A4   40 8F      90            1D58  5059          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)      ; Master Clear device
                                1D5D  5060          ASSUME  UCB$B_XD_OUTTIM EQ UCB$B_XD_INTIM+1
      02A5 C5      B4            1D5D  5061          CLRW    UCB$B_XD_INTIM(R5)              ; Stop all timers
                                1D61  5062          ASSUME  UCB$B_XD_DUP EQ UCB$B_XD_PRO+1
                                1D61  5063          ASSUME  UCB$B_XD_CON EQ UCB$B_XD_DUP+1
                                1D61  5064          ASSUME  UCB$B_XD_BFN EQ UCB$B_XD_CON+1
      02A8 C5      D4            1D61  5065          CLRL    UCB$B_XD_PRO(R5)                ; Reset all line characteristics
      57   1F      9A            1D65  5066          MOVZBL  S^#<MAX_DMP_TRIB-1>,R7          ; Set number of tributaries
59  0218 C547      D0            1D68  5067 30$:     MOVL    UCB$L_XD_CDB_VEC(R5)[R7],R9     ; Get next CDB address
           1A      13            1D6E  5068          BEQL    40$                             ; Br if none
     0218 C547     D4            1D70  5069          CLRL    UCB$L_XD_CDB_VEC(R5)[R7]        ; Remove CDB from vector
     0158 C547     D4            1D75  5070          CLRL    UCB$L_XD_PID_VEC(R5)[R7]        ; Zero PID entry
     01D8 C547     B4            1D7A  5071          CLRW    UCB$W_XD_CHAN_VEC(R5)[R7]       ; Zero channel entry
     0138 C547     94            1D7F  5072          CLRB    UCB$B_XD_TRIB_VEC(R5)[R7]       ; Reset trib address
          FED1     30            1D84  5073          BSBW    SHUTDOWN_TRIBT                  ; Abort all I/O on CDB
          00D6     30            1D87  5074          BSBW    CLEAR_CDB                       ; Deallocate CDB and restore quota
        DB 57      F4            1D8A  5075 40$:     SOBGEQ  R7,30$                          ; Loop if more tribs
                                1D8D  5076 ;
                                1D8D  5077 ; Clear device status flags
                                1D8D  5078 ;
      0134 C5      94            1D8D  5079          CLRB    UCB$B_XD_TRB_CNT(R5)            ; No tributaries
                                1D91  5080          DSBINT  UCB$B_DIPL(R5)                  ; Sync access to UCB
           22      AA            1D98  5081          BICW    #UCB$M_INT!UCB$M_POWER,-        ; Reset device status flags
        64 A5                   1D9A  5082                  UCB$W_STS(R5)                   ;
```

```
                FFF7 8F    AA   1D9C  5083          BICW    #^C<UCB$M_XD_FORK_PEND>,-        ; Reset all but fork pending
                  68 A5         1DA0  5084                  UCB$W_DEVSTS(R5)                 ; ..bit.
                                1DA2  5085          ENBINT                                   ; Restore IPL
                                1DA5  5086 ;
                                1DA5  5087 ; Release the receive and transmit buffer map registers
                                1DA5  5088 ;
                  54    24 A5   D0   1DA5  5089          MOVL    UCB$L_CRB(R5),R4                 ; Get CRB address
                              57    D4   1DA9  5090          CLRL    R7                               ; Init slot number
                                         1DAB  5091          ASSUME  UCB$L_XD_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XD_XMT_MAP
                56    00C4 C5    9E   1DAB  5092          MOVAB   UCB$L_XD_RCV_MAP(R5),R6          ; Get address of mapping slots
                  34 A4    86   D0   1DB0  5093 50$:      MOVL    (R6)+,CRB$L_INTD+VEC$W_MAPREG(R4) ; Set mapping info.
                              0A   19   1DB4  5094          BLSS    60$                              ; Br if none allocated
                                         1DB6  5095          RELMPR                                   ; Release the map registers
                  FC A6    01    CE   1DBC  5096          MNEGL   #1,-4(R6)                        ; Reset mapping info
                                         1DC0  5097 60$:      CLRBIT  R7,UCB$B_XD_RCV_MAP(R5)          ; Clear mapping slot flag
                  E6 57    0E    F2   1DC6  5098          AOBLSS  #MAX_RCV+MAX_XMT,R7,50$          ; Loop if more map registers
                                         1DCA  5099 ;
                                         1DCA  5100 ; Deallocate all receive buffers and complete all I/O request packets
                                         1DCA  5101 ;
                56    0094 C5    9E   1DCA  5102 70$:      MOVAB   UCB$Q_XD_QUEUES(R5),R6           ; Get address of first queue listhea
                    57    06    3C   1DCF  5103          MOVZWL  #UCB$C_XD_QUEUES,R7              ; Set number of queues
                  53    00 B6    0F   1DD2  5104 80$:      REMQUE  @(R6),R3                         ; Get next IRP/buffer
                              29   1D   1DD6  5105          BVS     120$                             ; Br if none - queue empty
                  0A    0A A3    91   1DD8  5106          CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP      ; Is this an IRP?
                              0A   13   1DDC  5107          BEQL    90$                              ; Br if yes
                  17    0A A3    91   1DDE  5108          CMPB    IRP$B_TYPE(R3),S^#DYN$C_NET      ; Is it a receive buffer?
                              0C   13   1DE2  5109          BEQL    110$                             ; Br if yes
                                         1DE4  5110          BUG_CHECK NOBUFPCKT,FATAL                ; Else, fatal error
                                         1DE8  5111 ;
                                         1DE8  5112 ; IRP
                                         1DE8  5113 ;
                        44 A3    D4   1DE8  5114 90$:      CLRL    IRP$L_CDB(R3)                    ; Definitely no CDB
                         FDDF    30   1DEB  5115          BSBW    ABORT_PKT                        ; Abort the I/O request
                           E2    11   1DEE  5116          BRB     80$                              ; Get next entry
                                         1DF0  5117 ;
                                         1DF0  5118 ; Receive buffer
                                         1DF0  5119 ;
                      50    53    D0   1DF0  5120 110$:     MOVL    R3,R0                            ; Set receive buffer address
                        42 A5    A0   1DF3  5121          ADDW    UCB$W_DEVBUFSIZ(R5),-            ; Restore quota
                      0136 C5         1DF6  5122                  UCB$W_XD_QUOTA(R5)               ;
                00000000'GF    16   1DF9  5123          JSB     G^COM$DRVDEALMEM                 ; Deallocate the receive buffer
                           D1    11   1DFF  5124          BRB     80$                              ; Get next entry
                                         1E01  5125 ;
                                         1E01  5126 ; Loop to next queue
                                         1E01  5127 ;
                      56    08    C0   1E01  5128 120$:     ADDL    #8,R6                            ; Skip to next queue listhead
                        CB 57    F5   1E04  5129          SOBGTR  R7,80$                           ; Loop if more queues
                                         1E07  5130 ;
                                         1E07  5131 ; Restore the buffered I/O quota to the starter
                                         1E07  5132 ;
                   50    02A0 C5    3C   1E07  5133          MOVZWL  UCB$L_XD_PID(R5),R0              ; Get process index of starter
                51    00000000'GF    D0   1E0C  5134          MOVL    G^SCH$GL_PCBVEC,R1               ; Get address of PCB address vector
                      50    6140    D0   1E13  5135          MOVL    (R1)[R0],R0                      ; Get PCB address of starter
                02A0 C5    60 A0    D1   1E17  5136          CMPL    PCB$L_PID(R0),UCB$L_XD_PID(R5)   ; Still the same process?
                              16   12   1E1D  5137          BNEQ    130$                             ; Br if not
                   50    0080 C0    D0   1E1F  5138          MOVL    PCB$L_JIB(R0),R0                 ; Get JIB address
                   51    0136 C5    3C   1E24  5139          MOVZWL  UCB$W_XD_QUOTA(R5),R1            ; Convert quota to longword
```

```
                20 A0    51    C0  1E29  5140          ADDL    R1,JIB$L_BYTCNT(R0)             ; Return byte count quota
                24 A0    51    C0  1E2D  5141          ADDL    R1,JIB$L_BYTLM(R0)              ; ..and limit
                   0136  C5    B4  1E31  5142          CLRW    UCB$W_XD_QUOTA(R5)              ; Reset quota
                   02D8  8F    BA  1E35  5143 130$:    POPR    #^M<R3,R4,R6,R7,R9>             ; Restore registers
                               05  1E39  5144          RSB                                     ; Return to caller
```

```
                        1E3A  5146          .SBTTL  REG_DUMP,  Device register dump routine
                        1E3A  5147
                        1E3A  5148 ;++
                        1E3A  5149 ; REG_DUMP, Dumps the contents of device registers to a buffer
                        1E3A  5150 ;
                        1E3A  5151 ; Functional description:
                        1E3A  5152 ;
                        1E3A  5153 ; Writes the number of longwords returned, and the contents of the
                        1E3A  5154 ; device registers into a diagnostic or error buffer.
                        1E3A  5155 ;
                        1E3A  5156 ; Inputs:
                        1E3A  5157 ;
                        1E3A  5158 ;       R0 = Diagnostic buffer address
                        1E3A  5159 ;       R5 = UCB address
                        1E3A  5160 ;
                        1E3A  5161 ;       IPL = FIPL
                        1E3A  5162 ;
                        1E3A  5163 ; Outputs:
                        1E3A  5164 ;
                        1E3A  5165 ;       R0 is destroyed.
                        1E3A  5166 ;
                        1E3A  5167 ;       R5 is preserved.
                        1E3A  5168 ;
                        1E3A  5169 ;--
                        1E3A  5170
                        1E3A  5171 REG_DUMP:                                    ; Dump device registers
     80   03    9A      1E3A  5172          MOVZBL  #3,(R0)+                   ; Store # of longwords returned
                        1E3D  5173          ASSUME  UCB$L_XD_SEL4 EQ UCB$L_XD_SEL0+4
80   02C6 C5    7D      1E3D  5174          MOVQ    UCB$L_XD_SEL0(R5),(R0)+    ; Store all CSR registers
  80   44 A5    D0      1E42  5175          MOVL    UCB$L_DEVDEPEND(R5),(R0)+  ; Store device dependent longword
                 05      1E46  5176          RSB                                ; Return
```

```
                                1E47  5178          .SBTTL  RETURN_IRP,  Deallocate IRP
                                1E47  5179          .SBTTL  RETURN_CDB,  Deallocate CDB
                                1E47  5180          .SBTTL  CLEAR_CDB,  Deallocate CDB
                                1E47  5181
                                1E47  5182 ;++
                                1E47  5183 ; RETURN_IRP - Deallocate IRP (Called from IOPOST)
                                1E47  5184 ; RETURN_CDB - Deallocate CDB (Called from IOPOST)
                                1E47  5185 ; CLEAR_CDB - Deallocate CDB (Called from CANCEL or the Fork Process)
                                1E47  5186 ;
                                1E47  5187 ; Functional description:
                                1E47  5188 ;
                                1E47  5189 ; These subroutines are called to deallocate the CDB and restore the byte
                                1E47  5190 ; count quota.
                                1E47  5191 ;
                                1E47  5192 ; Inputs:
                                1E47  5193 ;
                                1E47  5194 ;       R5 = IRP address (RETURN_IRP entry only)
                                1E47  5195 ;       R5 = CDB address (RETURN_CDB entry only)
                                1E47  5196 ;       R9 = CDB address (CLEAR_CDB entry only)
                                1E47  5197 ;
                                1E47  5198 ;       IPL = FIPL
                                1E47  5199 ;
                                1E47  5200 ; Outputs:
                                1E47  5201 ;
                                1E47  5202 ;       R0-R2 are destroyed.
                                1E47  5203 ;       R9 destroyed if CLEAR_CDB entry.
                                1E47  5204 ;
                                1E47  5205 ;--
                                1E47  5206
                                1E47  5207 RETURN_IRP:                                  ; Deallocate the IRP
          50    55    D0        1E47  5208          MOVL    R5,R0                       ; Copy IRP address
    00000000'GF    17           1E4A  5209          JMP     G^COM$DRVDEALMEM            ; Deallocate the IRP
                                1E50  5210
                                1E50  5211 RETURN_CDB:                                  ; Deallocate the CDB
                59    DD        1E50  5212          PUSHL   R9                          ; Save R9
             59    55    D0     1E52  5213          MOVL    R5,R9                       ; Copy CDB address
       48 A5    38 A5    D0     1E55  5214          MOVL    IRP$L_MEDIA(R5),CDB_L_PID(R5) ; Move PID to proper place
                04    10        1E5A  5215          BSBB    CLEAR_CDB                   ; Deallocate CDB
                59 8ED0         1E5C  5216          POPL    R9                          ; Restore R9
                      05        1E5F  5217          RSB                                 ; Return to caller (IOPOST)
                                1E60  5218
                                1E60  5219 CLEAR_CDB:                                   ; Deallocate CDB
          50    59    D0        1E60  5220          MOVL    R9,R0                       ; Copy CDB address
        59    48 A0    D0     1E63  5221          MOVL    CDB_L_PID(R0),R9            ; Get PID
    000C0000'GF    16           1E67  5222          JSB     G^COM$DRVDEALMEM            ; Deallocate the CDB
          50    59    3C        1E6D  5223          MOVZWL  R9,R0                       ; Get process index
    51  00000000'GF    D0        1E70  5224          MOVL    G^SCH$GL_PCBVEC,R1          ; Get address of PCB vector
        51    6140    D0        1E77  5225          MOVL    (R1)[R0],R1                 ; Get PCB address
        59    60 A1    D1        1E7B  5226          CMPL    PCB$L_PID(R1),R9            ; Is this the same PID?
                15    12        1E7F  5227          BNEQ    10$                         ; Br if not - just leave
      51    0080 C1    D0        1E81  5228          MOVL    PCB$L_JIB(R1),R1            ; Get JIB address
20 A1    00000060 8F    C0     1E86  5229          ADDL    #CDB_C_LENGTH,JIB$L_BYTCNT(R1) ; Restore byte count quota
24 A1    00000060 8F    C0     1E8E  5230          ADDL    #CDB_C_LENGTH,JIB$L_BYTLM(R1) ; ..and limit
                      05        1E96  5231 10$:     RSB                                 ; Return to caller
```

```
                                                I 11
                                                1E97  5233          .SBTTL  FIND_SLOT,  Find an empty slot in translation vector
                                                1E97  5234
                                                1E97  5235  ;++
                                                1E97  5236  ; FIND_SLOT - Find an empty slot in the translation vector
                                                1E97  5237  ;
                                                1E97  5238  ; Functional Description:
                                                1E97  5239  ;
                                                1E97  5240  ; This routine is called to return the index of the first available
                                                1E97  5241  ; slot in the translation vector starting from the hash point.
                                                1E97  5242  ;
                                                1E97  5243  ; Inputs:
                                                1E97  5244  ;
                                                1E97  5245  ;       R3 = IRP address
                                                1E97  5246  ;       R5 = UCB address
                                                1E97  5247  ;
                                                1E97  5248  ; Outputs:
                                                1E97  5249  ;
                                                1E97  5250  ;       R0 = status return
                                                1E97  5251  ;       R1,R2 = destroyed.
                                                1E97  5252  ;
                                                1E97  5253  ; If success:
                                                1E97  5254  ;
                                                1E97  5255  ;       IRP$B_INDEX(R3) = Index of empty slot in translation vector.
                                                1E97  5256  ;
                                                1E97  5257  ;--
                                                1E97  5258
                                                1E97  5259  FIND_SLOT:                                       ; Find empty slot in vector
      51   0C A3   05   00   EF                 1E97  5260          EXTZV   #0,#5,IRP$L_PID(R3),R1          ; Get start hash
           52   E2B0 CF   3C                    1E9D  5261          MOVZWL  W^MAX_W_TRIB,R2                 ; Get size of vector
           50   E2AB CF   3C                    1EA2  5262          MOVZWL  MAX_W_TRIB,R0                   ; Get number of tribs
                50   50   D2                    1EA7  5263          MCOML   R0,R0                           ; Get complement for modulo
                                                1EAA  5264
                     0158 C541   D5             1EAA  5265  10$:    TSTL    UCB$L_XD_PID_VEC(R5)[R1]        ; Empty slot?
                             0B   13            1EAF  5266          BEQL    20$                             ; Br if yes
                             51   D6            1EB1  5267          INCL    R1                              ; Set to next slot
                        51   50   CA            1EB3  5268          BICL    R0,R1                           ; Modulo the trib count
                          F1 52   F5            1EB6  5269          SOBGTR  R2,10$                          ; Br if more
                             50   D4            1EB9  5270          CLRL    R0                              ; Indicate error
                                  05            1EBB  5271          RSB                                     ; Return in error
                                                1EBC  5272  ;
                                                1EBC  5273  ; Empty slot found - save in IRP
                                                1EBC  5274  ;
      41 A3    51   90                          1EBC  5275  20$:    MOVB    R1,IRP$B_INDEX(R3)              ; Save index
           50   01   3C                        1EC0  5276          MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                                  05            1EC3  5277          RSB                                     ; Return to caller
```

```
                                1EC4  5279          .SBTTL  FIND_TRIB,  Find CDB from trib address
                                1EC4  5280
                                1EC4  5281 ;++
                                1EC4  5282 ; FIND_TRIB - Find CDB from trib address
                                1EC4  5283 ;
                                1EC4  5284 ; Functional Description:
                                1EC4  5285 ;
                                1EC4  5286 ; This routine is called to return the CDB address for a given tributary
                                1EC4  5287 ; address. The tributary address vector is scanned to make the comparison.
                                1EC4  5288 ;
                                1EC4  5289 ; Inputs:
                                1EC4  5290 ;
                                1EC4  5291 ;       R0 = tributary address
                                1EC4  5292 ;       R5 = UCB address
                                1EC4  5293 ;
                                1EC4  5294 ; Outputs:
                                1EC4  5295 ;
                                1EC4  5296 ;       R0 = status return
                                1EC4  5297 ;
                                1EC4  5298 ; If Success:
                                1EC4  5299 ;
                                1EC4  5300 ;       R9 = CDB address
                                1EC4  5301 ;
                                1EC4  5302 ;--
                                1EC4  5303
                                1EC4  5304 FIND_TRIB:                                   ; Find CDB from trib address
                                1EC4  5305          PUSHQ   R1                          ; Save R1, R2
          52   50   D0          1EC7  5306          MOVL    R0,R2                       ; Copy trib address
0138 C5   E282 CF   52   3A     1ECA  5307          LOCC    R2,MAX_W_TRIB,UCB$B_XD_TRIB_VEC(R5) ; Find trib in vector
                    16   13     1ED2  5308          BEQL    10$                         ; Leave if not found (R0 = 0)
                                1ED4  5309
          52   0138 C5   9E     1ED4  5310          MOVAB   UCB$B_XD_TRIB_VEC(R5),R2    ; Get address of trib vector
               51   52   C2     1ED9  5311          SUBL2   R2,R1                       ; Calculate index
     50   0218 C541   D0        1EDC  5312          MOVL    UCB$L_XD_CDB_VEC(R5)[R1],R0 ; Get CDB address
                    06   13     1EE2  5313          BEQL    10$                         ; Leave if none
               59   50   D0     1EE4  5314          MOVL    R0,R9                       ; Return CDB address
               50   01   3C     1EE7  5315          MOVZWL  S^#SS$_NORMAL,R0            ; Set success
                               1EEA  5316 10$:      POPQ    R1                          ; Restore R1, R2
                         05    1EED  5317          RSB                                 ; Return to caller
```

```
                                1EEE  5319          .SBTTL  XLATE,  Translate PID and Channel to CDB address
                                1EEE  5320          .SBTTL  XLATE_ALT,  Translate PID and Channel to CDB address
                                1EEE  5321
                                1EEE  5322 ;++
                                1EEE  5323 ; XLATE - Translate PID and Channel to CDB address
                                1EEE  5324 ; XLATE_ALT - Translate PID and Channel to CDB address
                                1EEE  5325 ;
                                1EEE  5326 ; Functional Description:
                                1EEE  5327 ;
                                1EEE  5328 ; This routine is called to return the CDB address for a particular
                                1EEE  5329 ; channel. If the controller is running in point to point mode or
                                1EEE  5330 ; the controller is a tributary then it is assumed that the first
                                1EEE  5331 ; hash into the CDB address vector has the one and only CDB address.
                                1EEE  5332 ;
                                1EEE  5333 ; This routine hashes the low order 5 bits of the PID to start the
                                1EEE  5334 ; search. This should optimize scanning of the PID vector only if
                                1EEE  5335 ; the CDB already exists.
                                1EEE  5336 ;
                                1EEE  5337 ; Inputs:
                                1EEE  5338 ;
                                1EEE  5339 ;       R0 = PID (XLATE_ALT only)
                                1EEE  5340 ;       R3 = IRP address
                                1EEE  5341 ;       R5 = UCB address
                                1EEE  5342 ;
                                1EEE  5343 ; Implicit Inputs:
                                1EEE  5344 ;
                                1EEE  5345 ;       IRP$L_PID(R3) = PID of requestor (XLATE only)
                                1EEE  5346 ;       IRP$W_CHAN(R3) = Channel of requestor
                                1EEE  5347 ;
                                1EEE  5348 ; Outputs:
                                1EEE  5349 ;
                                1EEE  5350 ;       R0 = status return for success of call.
                                1EEE  5351 ;       R9 = CDB address (if success, else zero)
                                1EEE  5352 ;
                                1EEE  5353 ;       R1,R2 are destroyed.
                                1EEE  5354 ;
                                1EEE  5355 ;--
                                1EEE  5356
                                1EEE  5357 XLATE:                                       ; Translate IRP into CDB address
          50    0C A3    D0     1EEE  5358          MOVL    IRP$L_PID(R3),R0             ; Get PID
                                1EF2  5359 XLATE_ALT:                                   ; Alternate translation entry
               53    DD         1EF2  5360          PUSHL   R3                           ; Save IRP address
          53   28 A3    3C     1EF4  5361          MOVZWL  IRP$W_CHAN(R3),R3            ; Get channel
                17    10         1EF8  5362          BSBB    XLATE_PID_CHAN               ; Translate PID and channel
               53 8ED0    1EFA  5363          POPL    R3                           ; Restore IRP address
             10 50    E9     1EFD  5364          BLBC    R0,10$                       ; Br if error
                                1F00  5365 ;
                                1F00  5366 ; Found PID and Channel in translation vector, get CDB address
                                1F00  5367 ;
          41 A3    51    90     1F00  5368          MOVB    R1,IRP$B_INDEX(R3)           ; Save index in IRP
 44 A3     0218 C541    DE     1F04  5369          MOVAL   UCB$L_XD_CDB_VEC(R5)[R1],-   ; Store address of CDB address
                                1F0B  5370                  IRP$L_CDB(R3)                ;
             0E A9    90     1F0B  5371          MOVB    CDB_B_TRB_ADDR(R9),-         ; Store trib address also
             40 A3           1F0E  5372                  IRP$Q_STATION(R3)            ;
                05    05     1F10  5373 10$:     RSB                                  ; Return to caller
                                1F11  5374 ;+
                                1F11  5375 ; The Following entry point is called by the CANCEL routine also.
```

```
                                1F11  5376 ;
                                1F11  5377 ; Inputs:
                                1F11  5378 ;
                                1F11  5379 ;       R0 = PID
                                1F11  5380 ;       R3 = Channel for request
                                1F11  5381 ;
                                1F11  5382 ; Outputs:
                                1F11  5383 ;
                                1F11  5384 ;       R0 = Status return for request
                                1F11  5385 ;       R9 = CDB address if success, else zero
                                1F11  5386 ;-
                                1F11  5387 XLATE_PID_CHAN:                                      ; Translate PID & CHAN into CDB addr
51   50   05   00   EF          1F11  5388         EXTZV   #0,#5,R0,R1                         ; Hash PID into 32 indices
    29 68 A5   02   E0          1F16  5389         BBS     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),40$ ; Br if point to point
                                1F1B  5390 ;       BBC     #XM$V_CHR_CTRL,UCB$[_DEVDEPEND(R5),40$ ; Br if not control station
         52   51   D0          1F1B  5391         MOVL    R1,R2                               ; Save start hash
   0158 C541   50   D1          1F1E  5392 10$:    CMPL    R0,UCB$L_XD_PID_VEC(R5)[R1]         ; PIDs match?
                    08   12     1F24  5393         BNEQ    20$                                 ; Br if not
   01D8 C541   53   B1          1F26  5394         CMPW    R3,UCB$W_XD_CHAN_VEC(R5)[R1]        ; Channels match?
                    16   13     1F2C  5395         BEQL    40$                                 ; Br if yes - got it
                    51   D6     1F2E  5396 20$:    INCL    R1                                  ; Else, set to next entry
 51    FFFFFFE0 8F   CA          1F30  5397         BICL    #^C<31>,R1                          ; Modulo 32
         52   51   D1          1F37  5398         CMPL    R1,R2                               ; Back to start?
                    E2   12     1F3A  5399         BNEQ    10$                                 ; Br if more to check
      50   0084 8F   3C          1F3C  5400 30$:    MOVZWL  #SS$_DEVOFFLINE,R0                  ; Return channel offline
                    59   D4     1F41  5401         CLRL    R9                                  ; Return zero in R9
                    05   1F43  5402         RSB                                             ; Return to caller
                                1F44  5403 ;
                                1F44  5404 ; Found match on PID and Channel
                                1F44  5405 ;
59   0218 C541   D0          1F44  5406 40$:    MOVL    UCB$L_XD_CDB_VEC(R5)[R1],R9         ; Get CDB address
                    F0   13     1F4A  5407         BEQL    30$                                 ; Br if no CDB address - error
      50   01   3C          1F4C  5408         MOVZWL  S^#SS$_NORMAL,R0                    ; Set successful return status
                    05   1F4F  5409         RSB                                             ; Return
```

```
                     1F50  5411          .SBTTL  GET_CDB,  Get CDB address from IRP
                     1F50  5412
                     1F50  5413 ;++
                     1F50  5414 ; GET_CDB - Get CDB address from IRP
                     1F50  5415 ;
                     1F50  5416 ; Inputs:
                     1F50  5417 ;
                     1F50  5418 ;       R3 = IRP address
                     1F50  5419 ;       R5 = UCB address
                     1F50  5420 ;
                     1F50  5421 ; Implicit Inputs:
                     1F50  5422 ;
                     1F50  5423 ;       IRP$L_CDB(R3) = address of CDB address (pointer into UCB vector)
                     1F50  5424 ;
                     1F50  5425 ; Outputs:
                     1F50  5426 ;
                     1F50  5427 ;       R0 = status of request
                     1F50  5428 ;       R9 = CDB address if present in UCB vector
                     1F50  5429 ;
                     1F50  5430 ;--
                     1F50  5431
                     1F50  5432 GET_CDB:                                     ; Get CDB address from IRP
59   44 A3   D0      1F50  5433          MOVL    IRP$L_CDB(R3),R9           ; Get address of CDB address
        04   12      1F54  5434          BNEQ    10$                        ; Br if present
                     1F56  5435          BUG_CHECK NOBUFPCKT,FATAL          ; Else, fatal error
                     1F5A  5436
   50   01   3C      1F5A  5437 10$:     MOVZWL  S^#SS$_NORMAL,R0           ; Assume good CDB address
   59   69   D0      1F5D  5438          MOVL    (R9),R9                    ; Get CDB address
        03   12      1F60  5439          BNEQ    20$                        ; Br if good
   50   2C   3C      1F62  5440          MOVZWL  #SS$_ABORT,R0              ; Else, abort the request
        05   1F65  5441 20$:     RSB                                        ; Return to caller
```

```
                        1F66  5443          .SBTTL  CHG_UCB,  Change UCB parameter values
                        1F66  5444
                        1F66  5445 ;++
                        1F66  5446 ; CHG_UCB - Change UCB parameter values
                        1F66  5447 ;
                        1F66  5448 ; Functional description:
                        1F66  5449 ;
                        1F66  5450 ; This routine is called to initialize the UCB with new P1 and P2 buffer
                        1F66  5451 ; characteristics. It is assumed here that the parameters have already
                        1F66  5452 ; been validated.
                        1F66  5453 ;
                        1F66  5454 ; Inputs:
                        1F66  5455 ;
                        1F66  5456 ;       R3 = IRP address
                        1F66  5457 ;       R5 = UCB address
                        1F66  5458 ;
                        1F66  5459 ;       IPL = FIPL
                        1F66  5460 ;
                        1F66  5461 ; Outputs:
                        1F66  5462 ;
                        1F66  5463 ;       R0-R2 = destroyed.
                        1F66  5464 ;
                        1F66  5465 ;--
                        1F66  5466
                        1F66  5467 CHG_UCB:                                     ; Change UCB parameters
          54    DD      1F66  5468          PUSHL   R4                        ; Save R4
51 38 A3    E9          1F68  5469          BLBC    IRP$L_MEDIA(R3),10$       ; Br if no P1 buffer
                        1F6C  5470 ;
                        1F6C  5471 ; Set new P1 buffer characteristics
                        1F6C  5472 ;
      05 68 A5    00    E0    1F6C  5473          BBS     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),5$ ; Br if device inited
      42 A5     3A A3     B0    1F71  5474          MOVW    IRP$L_MEDIA+2(R3),UCB$W_DEVBUFSIZ(R5) ; Set new buffer size
                        1F76  5475 5$:      ASSUME  P1_LINE_MASK EQ <^XFF>
44 A5   18    08    00    F0    1F76  5476          INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5)   ; Reset all Read/Write flags
      44 A5     3C A3     C8    1F7C  5477          BISL    IRP$L_MEDIA+4(R3),UCB$L_DEVDEPEND(R5) ; Set new characteristics
                        1F81  5478 ;
                        1F81  5479 ; Now update UCB based on P1 buffer
                        1F81  5480 ;
                        1F81  5481          ASSUME  NMA$C_LINPR_POI EQ 0
            02A8 C5     94    1F81  5482          CLRB    UCB$B_XD_PRO(R5)          ; Assume point to point mode
      44 A5     E0 8F     93    1F85  5483          BITB    #XM$M_CHR_DMC!XM$M_CHR_TRIB!- ; Is this point to point mode?
                        1F8A  5484                  XM$M_CHR_CTRL,UCB$L_DEVDEPEND(R5) ;
                  17    13    1F8A  5485          BEQL    6$                        ; Br if yes
                        1F8C  5486          ASSUME  NMA$C_LINPR_CON EQ 1
            02A8 C5     96    1F8C  5487          INCB    UCB$B_XD_PRO(R5)          ; Assume control station
                  06    E0    1F90  5488          BBS     #XM$V_CHR_CTRL,-          ; Br if control mode
               0E 44 A5         1F92  5489                  UCB$L_DEVDEPEND(R5),6$    ;
                        1F95  5490          ASSUME  NMA$C_LINPR_TRI EQ 2
            02A8 C5     96    1F95  5491          INCB    UCB$B_XD_PRO(R5)          ; Assume trib station
                  07    E0    1F99  5492          BBS     #XM$V_CHR_TRIB,-          ; Br if trib station
               05 44 A5         1F9B  5493                  UCB$L_DEVDEPEND(R5),6$    ;
            02A8 C5     04    90    1F9E  5494          MOVB    #NMA$C_LINPR_DMC,UCB$B_XD_PRO(R5) ; Must be DMC compatible mode
                        1FA3  5495 6$:      ASSUME  NMA$C_LINCN_NOR EQ 0
            02AA C5     94    1FA3  5496          CLRB    UCB$B_XD_CON(R5)          ; Assume normal mode
                  01    E1    1FA7  5497          BBC     #XM$V_CHR_LOOPB,-         ; Br if not loopback mode
               04 44 A5         1FA9  5498                  UCB$L_DEVDEPEND(R5),7$    ;
                        1FAC  5499          ASSUME  NMA$C_LINCN_LOO EQ 1
```

```
                02AA C5   96  1FAC  5500          INCB    UCB$B_XD_CON(R5)                ; Must be loopback mode
                          1FB0  5501 7$:        ASSUME  NMA$C_DPX_FUL EQ 0
                02A9 C5   94  1FB0  5502          CLRB    UCB$B_XD_DUP(R5)                ; Assume full duplex
                     02   E1  1FB4  5503          BBC     #XM$V_CHR_HDPLX,-               ; Br if not half duplex
                04 44 A5       1FB6  5504                  UCB$L_DEVDEPEND(R5),10$         ;
                          1FB9  5505          ASSUME  NMA$C_DPX_HAL EQ 1
                02A9 C5   96  1FB9  5506          INCB    UCB$B_XD_DUP(R5)                ; Must be half duplex
                          1FBD  5507
                          1FBD  5508 ;
                          1FBD  5509 ; Set new P2 buffer characteristics
                          1FBD  5510 ;
                54   55   D0  1FBD  5511 10$:     MOVL    R5,R4                           ; Copy UCB address
           51   E0B4 CF   9E  1FC0  5512          MOVAB   LINE_PARAM,R1                   ; Get address of verification table
                   01B0   30  1FC5  5513          BSBW    UPDATE_P2                       ; Update the UCB
                          1FC8  5514 ;
                          1FC8  5515 ; Set device characteristics and device mode definition
                          1FC8  5516 ;
                51   01   9A  1FC8  5517          MOVZBL  S^#1,R1                         ; Assume point to point mode
                  44 A5   94  1FCB  5518          CLRB    UCB$L_DEVDEPEND(R5)             ; Reset all characteristics
                          1FCE  5519          ASSUME  NMA$C_LINPR_POI EQ 0
           52   02A8 C5   9A  1FCE  5520          MOVZBL  UCB$B_XD_PRO(R5),R2             ; Get protocol mode
                     21   13  1FD3  5521          BEQL    30$                             ; Br if point to point mode
                          1FD5  5522
                          1FD5  5523          ASSUME  NMA$C_LINPR_CON EQ 1
                     51   D6  1FD5  5524          INCL    R1                              ; Assume control station mode
                     52   D7  1FD7  5525          DECL    R2                              ; Control station mode?
                     07   12  1FD9  5526          BNEQ    15$                             ; Br if not
                          1FDB  5527          SETBIT  #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5) ; Indicate control station
                     14   11  1FE0  5528          BRB     30$                             ;
                          1FE2  5529
                          1FE2  5530 15$:     ASSUME  NMA$C_LINPR_TRI EQ 2
                     52   D7  1FE2  5531          DECL    R2                              ; Tributary station protocol?
                     09   12  1FE4  5532          BNEQ    20$                             ; Br if no
                     51   D6  1FE6  5533          INCL    R1                              ; Set mode to trib station
                          1FE8  5534          SETBIT  #XM$V_CHR_TRIB,UCB$L_DEVDEPEND(R5) ; Indicate tributary station
                     07   11  1FED  5535          BRB     30$                             ; And continue
                          1FEF  5536
                          1FEF  5537 20$:     SETBIT  #XM$V_CHR_DMC,UCB$L_DEVDEPEND(R5) ; Must be DMC compatible mode
                     51   D4  1FF4  5538          CLRL    R1                              ; Set mode to DMC compatible
                          1FF6  5539
           51   51   01   78  1FF6  5540 30$:     ASHL    #1,R1,R1                        ; Shift for mode definition code
                          1FFA  5541          SETBIT  #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Assume half duplex mode
                          1FFF  5542          ASSUME  NMA$C_DPX_FUL EQ 0
                02A9 C5   95  1FFF  5543          TSTB    UCB$B_XD_DUP(R5)                ; Full duplex mode?
                     07   12  2003  5544          BNEQ    40$                             ; Br if no - okay
                     51   D6  2005  5545          INCL    R1                              ; Set new device mode
                          2007  5546          CLRBIT  #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Set to full duplex mode
           02A4 C5   51   90  200C  5547 40$:     MOVB    R1,UCB$B_XD_MODE(R5)            ; Save device mode (DMP format)
                          2011  5548          CLRBIT  #XM$V_CHR_LOOPB,UCB$L_DEVDEPEND(R5) ; Clear loopback bit
                          2016  5549          ASSUME  NMA$C_LINCN_NOR EQ 0
                02AA C5   95  2016  5550          TSTB    UCB$B_XD_CON(R5)                ; Is line in normal mode?
                     05   13  201A  5551          BEQL    50$                             ; Br if yes
                          201C  5552          SETBIT  #XM$V_CHR_LOOPB,UCB$L_DEVDEPEND(R5) ; Set loopback mode
                          2021  5553 50$:     CLRBIT  #UCB$V_XD_PTP,UCB$W_DEVSTS(R5)  ; Assume not pt-pt
            000000C0 8F   D3  2026  5554          BITL    #XM$M_CHR_CTRL!XM$M_CHR_TRIB,-  ; Is this trib or ctrl station?
                  44 A5       202C  5555                  UCB$L_DEVDEPEND(R5)             ;
                     05   12  202E  5556          BNEQ    60$                             ; Br if yes
```

```
                2030  5557          SETBIT  #UCB$V_XD_PTP,UCB$W_DEVSTS(R5)  ; Else, indicate pt-pt
54 8ED0         2035  5558 60$:     POPL    R4                              ; Restore R4
       05       2038  5559          RSB                                     ; Return to caller
```

```
                     2039  5561          .SBTTL  CHG_CDB,  Change CDB parameter values
                     2039  5562
                     2039  5563 ;++
                     2039  5564 ; CHG_CDB - Change CDB parameter values
                     2039  5565 ;
                     2039  5566 ; Functional description:
                     2039  5567 ;
                     2039  5568 ; This routine is called to initialize the CDB with new P1 and P2 buffer
                     2039  5569 ; characteristics. It is assumed here that the parameters have already
                     2039  5570 ; been validated.
                     2039  5571 ;
                     2039  5572 ; Inputs:
                     2039  5573 ;
                     2039  5574 ;       R3 = IRP address
                     2039  5575 ;       R5 = UCB address
                     2039  5576 ;       R9 = CDB address
                     2039  5577 ;
                     2039  5578 ;       IPL = FIPL
                     2039  5579 ;
                     2039  5580 ; Outputs:
                     2039  5581 ;
                     2039  5582 ;       R1,R2 = destroyed.
                     2039  5583 ;--
                     2039  5584
                     2039  5585 CHG_CDB:                                       ; Validate P2 buffer
          54    DD   2039  5586          PUSHL   R4                            ; Save R4
     1D 38 A3   E9   203B  5587          BLBC    IRP$L_MEDIA(R3),10$           ; Br if no P1 buffer
                     203F  5588 ;
                     203F  5589 ; Set new P1 buffer characteristics
                     203F  5590 ;
     FFFFD700 8F    CA   203F  5591          BICL    #^C<P1_TRIB_MASK>,-           ; Clear all but special
                 69       2045  5592                  CDB_L_DEVDEPEND(R9)           ; ..fields in CDB
     00002800 8F    CA   2046  5593          BICL    #XM$M_STS_ACTIVE!XM$M_STS_RUNNING,- ; Clear special fields
                 3C A3    204C  5594                  IRP$L_MEDIA+4(R3)             ; ..from IRP
      69    3C A3   C8   204E  5595          BISL    IRP$L_MEDIA+4(R3),CDB_L_DEVDEPEND(R9) ; Set new characteristics
                     2052  5596          ASSUME  NMA$C_STATE_ON EQ 0
                     2052  5597          ASSUME  NMA$C_STATE_OFF EQ 1
                4F A9    94   2052  5598          CLRB    CDB_B_MST(R9)         ; Assume MOP
      03 69     00    E0   2055  5599          BBS     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),10$ ; Br if true
                4F A9    96   2059  5600          INCB    CDB_B_MST(R9)         ; Else, NORMAL mode
                     205C  5601 ;
                     205C  5602 ; Set new P2 buffer characteristics
                     205C  5603 ;
          54    59   D0   205C  5604 10$:     MOVL    R9,R4                         ; Copy CDB address
     51   E067 CF   9E   205F  5605          MOVAB   TRIB_PARAM,R1                 ; Get address of verification table
                0111   30   2064  5606          BSBW    UPDATE_P2                     ; Update CDB
        51    0E A9   9A   2067  5607          MOVZBL  CDB_B_TRB_ADDR(R9),R1         ; Get trib address
        40 A3   51    90   206B  5608          MOVB    R1,IRP$Q_STATION(R3)          ; Return trib address
        52    41 A3   9A   206F  5609          MOVZBL  IRP$B_INDEX(R3),R2            ; Get vector index
  0138 C542    51    90   2073  5610          MOVB    R1,UCB$B_XD_TRIB_VEC(R5)[R2]  ; Set new trib address
                     2079  5611          CLRBIT  #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9) ; Assume not MOP mode
                     207D  5612          ASSUME  NMA$C_STATE_ON EQ 0
                     207D  5613          ASSUME  NMA$C_STATE_OFF EQ 1
             04 4F A9   E8   207D  5614          BLBS    CDB_B_MST(R9),20$             ; Br if MAINT state is OFF
                     2081  5615          SETBIT  #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9) ; Else, indicate MOP
                54 8ED0   2085  5616 20$:     POPL    R4                            ; Restore R4
                   05     2088  5617          RSB                                   ; Return to caller
```

```
                     2089  5619          .SBTTL  VALIDATE_P2,  Validate P2 buffer parameters
                     2089  5620          .SBTTL  VALIDATE_P2_CDB,  Validate P2 buffer with CDB
                     2089  5621          .SBTTL  VALIDATE_P2_UCB,  Validate P2 buffer with UCB
                     2089  5622
                     2089  5623 ;++
                     2089  5624 ; VALIDATE_P2 - Validate P2 buffer parameters
                     2089  5625 ; VALIDATE_P2_CDB - Validate P2 buffer with CDB
                     2089  5626 ; VALIDATE_P2_UCB - Validate P2 buffer with UCB
                     2089  5627 ;
                     2089  5628 ; This routine is called to validate the P2 buffer parameters. The parameters
                     2089  5629 ; are checked against a parameter table which verifies that the minimum value
                     2089  5630 ; and maximum value is not violated, and that invalid status flags are not set.
                     2089  5631 ;
                     2089  5632 ; Inputs:
                     2089  5633 ;
                     2089  5634 ;       R1 = Address of parameter verification table (VALIDATE_P2 entry only)
                     2089  5635 ;       R2 = Status word from UCB or CDB
                     2089  5636 ;       R3 = IRP address
                     2089  5637 ;       R4 = UCB or CDB address for value checking (VALIDATE_P2 entry only)
                     2089  5638 ;       R5 = UCB address
                     2089  5639 ;       R9 = CDB address (VALIDATE_P2_CDB entry only)
                     2089  5640 ;
                     2089  5641 ;       IPL = FIPL
                     2089  5642 ;
                     2089  5643 ; Outputs:
                     2089  5644 ;
                     2089  5645 ;       R0 = status return of parameters
                     2089  5646 ;
                     2089  5647 ; If no error:
                     2089  5648 ;       R1 = Address of parameter verification table
                     2089  5649 ; If error:
                     2089  5650 ;       R1 = Bad parameter value
                     2089  5651 ;
                     2089  5652 ;       All other registers are preserved.
                     2089  5653 ;
                     2089  5654 ;--
                     2089  5655          .ENABL  LSB
                     2089  5656 VALIDATE_P2_CDB:                                  ; Validate P2 buffer with CDB
          54    DD   2089  5657          PUSHL   R4                              ; Save R4
51   E03B CF    9E   208B  5658          MOVAB   TRIB_PARAM,R1                   ; Get address of verification table
     54   59    D0   2090  5659          MOVL    R9,R4                           ; Copy CDB address
          0A    11   2093  5660          BRB     10$                             ;
                     2095  5661 VALIDATE_P2_UCB:                                  ; Validate P2 buffer with UCB
          54    DD   2095  5662          PUSHL   R4                              ; Save R4
51   DFDD CF    9E   2097  5663          MOVAB   LINE_PARAM,R1                   ; Get address of verification table
     54   55    D0   209C  5664          MOVL    R5,R4                           ; Copy UCB address
          04    10   209F  5665 10$:     BSBB    VALIDATE_P2                     ; Do the validation
          54 8ED0    20A1  5666          POPL    R4                              ; Restore R4
             05      20A4  5667          RSB                                     ; Return to caller
                     20A5  5668          .DSABL  LSB
                     20A5  5669
                     20A5  5670 VALIDATE_P2:                                    ; Validate P2 buffer parameters
     03EE 8F    BB   20A5  5671          PUSHR   #^M<R1,R2,R3,R5,R6,R7,R8,R9>    ; Save registers
                     20A9  5672                                                  ; NB:R1 must be on top of stack
  56   2C A3    D0   20A9  5673          MOVL    IRP$L_SVAPTE(R3),R6             ; Get system P2 buffer address
          03    12   20AD  5674          BNEQ    10$                             ; Br if a system buffer
        00B6    31   20AF  5675          BRW     150$                            ; Else, leave
```

```
                                20B2  5676 ;
                                20B2  5677 ; Make sure that if trib address given, that it is unique, or belongs to this
                                20B2  5678 ; circuit.
                                20B2  5679 ;
                     59   52    D0  20B2  5680 10$:     MOVL    R2,R9                           ; Save status word
                51   0474 8F    3C  20B5  5681          MOVZWL  #NMA$C_PCCI_TRI,R1              ; Get trib address
                        0186    30  20BA  5682          BSBW    UNPACK_P2_BOF                   ; From P2 buffer
                       20 50    E9  20BD  5683          BLBC    R0,20$                          ; Br if no trib address
          0138 C5   E08C CF 52    3A  20C0  5684          LOCC    R2,MAX_W_TRIB,UCB$B_XD_TRIB_VEC(R5) ; Try to find same trib address
                           16    13  20C8  5685          BEQL    20$                             ; Br if unique
                58   0138 C5    9E  20CA  5686          MOVAB   UCB$B_XD_TRIB_VEC(R5),R8        ; Else, get address of trib vec
                     51   58    C2  20CF  5687          SUBL2   R8,R1                           ; Calculate index
                  51   41 A3    91  20D2  5688          CMPB    IRP$B_INDEX(R3),R1              ; If not unique, must be same CEB
                           08    13  20D6  5689          BEQL    20$                             ; Br if same circuit
                50   0474 8F    3C  20D8  5690          MOVZWL  #NMA$C_PCCI_TRI,R0              ; Return bad parameter
                        008D    31  20DD  5691          BRW     170$                            ; Else, error
                                20E0  5692
                     52   59    D0  20E0  5693 20$:     MOVL    R9,R2                           ; Restore status word
                  58   32 A3    3C  20E3  5694          MOVZWL  IRP$W_BCNT(R3),R8               ; Get size of P2 buffer
                     58   06    C6  20E7  5695          DIVL    #6,R8                           ; Calculate number of parameters
                     56   66    D0  20EA  5696          MOVL    P2B_L_POINTER(R6),R6            ; Point to start of P2 data
                                20ED  5697 ;
                                20ED  5698 ; Loop to check next parameter in P2 buffer
                                20ED  5699 ;
                     50   86    3C  20ED  5700 30$:     MOVZWL  (R6)+,R0                        ; Get parameter type from P2
                     55   86    D0  20F0  5701          MOVL    (R6)+,R5                        ; Get parameter value from P2
                     57   6E    D0  20F3  5702          MOVL    (SP),R7                         ; Get parameter table address
                                20F6  5703 ;
                                20F6  5704 ; Loop to check P2 buffer parameter to Line parameter table
                                20F6  5705 ;
                59   87    B0  20F6  5706 40$:     MOVW    (R7)+,R9                        ; Get parameter + flags
                     72    13  20F9  5707          BEQL    170$                            ; Br if end of verify table
          50   59   0C    00  ED  20FB  5708          CMPZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R0   ; Parameters match?
                     16    13  2100  5709          BEQL    50$                             ; Br if yes
                     87    B5  2102  5710          TSTW    (R7)+                           ; Skip offset word
                                2104  5711          SKIP    PRM_V_MIN,R9,R7                 ; Skip minimum value
                                210A  5712          SKIP    PRM_V_MAX,R9,R7                 ; Skip maximum value
                                2110  5713          SKIP    PRM_V_INVALID,R9,R7             ; Skip invalid flags
                     DE    11  2116  5714          BRB     40$                             ; Try next parameter
                                2118  5715 ;
                                2118  5716 ; Match found - nullify if same value & check min,max,valid,invalid
                                2118  5717 ;
                     51   87    B0  2118  5718 50$:     MOVW    (R7)+,R1                        ; Get offset + width
                     54   D5    211B  5719          TSTL    R4                              ; Is data structure present?
                     2B   13    211D  5720          BEQL    100$                            ; Br if not - check values
          53   51   02   0E    EF  211F  5721          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R3 ; Get width only
          51   51   0E   00    EF  2124  5722          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1 ; Get offset only
                     51   54    C0  2129  5723          ADDL    R4,R1                           ; Calculate address of datum
                                212C  5724          CASE    R3,TYPE=B,LIMIT=#1,<-           ; Br to handler
                                212C  5725                  60$,-                           ;   Byte value
                                212C  5726                  70$,-                           ;   Word value
                                212C  5727                  80$>                            ;   Longword value
                                2136  5728 ;
                                2136  5729 ; Byte value in structure
                                2136  5730 ;
                     61   55    91  2136  5731 60$:     CMPB    R5,(R1)                         ; Is this the same?
                     08    11  2139  5732          BRB     90$                             ; Check result
```

```
                     213B  5733 ;
                     213B  5734 ; Word value
                     213B  5735 ;
    61    55    B1   213B  5736 70$:    CMPW    R5,(R1)                 ; Is this the same?
          03    11   213E  5737         BRB     90$                     ; Check result
                     2140  5738
    61    55    D1   2140  5739 80$:    CMPL    R5,(R1)                 ; Is this the same?
          05    12   2143  5740 90$:    BNEQ    100$                    ; Br if no - continue checks
       FA A6    B4   2145  5741         CLRW    -6(R6)                  ; Nullify the parameter code
          1B    11   2148  5742         BRB     140$                    ; Try next parameter - skip checks
                     214A  5743
 05 59    0C    E1   214A  5744 100$:   BBC     #PRM_V_MIN,R9,110$      ; Br if no minimum value
    87    55    B1   214E  5745         CMPW    R5,(R7)+                ; Is the value too small?
          1A    1F   2151  5746         BLSSU   170$                    ; Br if yes - error
 05 59    0D    E1   2153  5747 110$:   BBC     #PRM_V_MAX,R9,130$      ; Br if no maximum value
    87    55    B1   2157  5748         CMPW    R5,(R7)+                ; Is the value too big?
          11    1A   215A  5749         BGTRU   170$                    ; Br if yes - error
 05 59    0E    E1   215C  5750 130$:   BBC     #PRM_V_INVALID,R9,140$  ; Br if no invalid flags
    52    87    B3   2160  5751         BITW    (R7)+,R2                ; Check invalid bits
          08    12   2163  5752         BNEQ    170$                    ; Br if on - error
       85 58    F5   2165  5753 140$:   SOBGTR  R8,30$                  ; Loop if more parameters
                     2168  5754
    50    01    3C   2168  5755 150$:   MOVZWL  S^#SS$_NORMAL,R0        ; Set success return
          06    11   216B  5756         BRB     180$                    ; And return
                     216D  5757
    6E    50    3C   216D  5758 170$:   MOVZWL  R0,(SP)                 ; Return bad parameter code
                     2170  5759                                         ; * R1 Must be on top of stack
    50    14    3C   2170  5760         MOVZWL  #SS$_BADPARAM,R0        ; Set error return
 03EE 8F    BA   2173  5761 180$:   POPR    #^M<R1,R2,R3,R5,R6,R7,R8,R9>    ; Restore registers
                05   2177  5762         RSB                             ; Return to caller
```

```
                     2178  5764          .SBTTL  UPDATE_P2,  Update UCB/CDB based on P2 buffer parameters
                     2178  5765
                     2178  5766 ;++
                     2178  5767 ; UPDATE_P2 - Update UCB/CDB with P2 buffer parameters
                     2178  5768 ;
                     2178  5769 ; This routine is called to update the UCB/CDB with the P2 buffer parameters.
                     2178  5770 ; The parameters are stored in the appropriate cells of the UCB/CDB.
                     2178  5771 ;
                     2178  5772 ; Inputs:
                     2178  5773 ;
                     2178  5774 ;       R1 = Address of parameter verification table
                     2178  5775 ;       R3 = IRP address
                     2178  5776 ;       R4 = UCB or CDB address for storing
                     2178  5777 ;       R5 = UCB address
                     2178  5778 ;
                     2178  5779 ;       IPL = FIPL
                     2178  5780 ;
                     2178  5781 ; Outputs:
                     2178  5782 ;
                     2178  5783 ;       R0 = destroyed.
                     2178  5784 ;       All other registers are preserved.
                     2178  5785 ;
                     2178  5786 ;--
                     2178  5787
                     2178  5788 UPDATE_P2:                                          ; Update the UCB/CDB
    03E6 8F    BB    2178  5789          PUSHR   #^M<R1,R2,R5,R6,R7,R8,R9>         ; Save registers
                     217C  5790                                                    ; NB:R1 must be on top of stack
56   2C A3    D0    217C  5791          MOVL    IRP$L_SVAPTE(R3),R6               ; Get system P2 buffer address
        62    13    2180  5792          BEQL    80$                               ; Br if no system buffer
58   32 A3    3C    2182  5793          MOVZWL  IRP$W_BCNT(R3),R8                 ; Get size of P2 buffer
   58   06    C6    2186  5794          DIVL    #6,R8                             ; Calculate number of parameters
   56   66    D0    2189  5795          MOVL    P2B_L_POINTER(R6),R6              ; Point to start of data
                     218C  5796 ;
                     218C  5797 ; Loop to get next parameter from P2 buffer
                     218C  5798 ;
          50   86    3C    218C  5799 10$:    MOVZWL  (R6)+,R0                          ; Get parameter type from P2
          55   86    D0    218F  5800          MOVL    (R6)+,R5                          ; Get parameter value from P2
          57   6E    D0    2192  5801          MOVL    (SP),R7                           ; Get parameter table address
                     2195  5802 ;
                     2195  5803 ; Loop to store buffer parameter in UCB/CDB
                     2195  5804 ;
          59   87    B0    2195  5805 20$:    MOVW    (R7)+,R9                          ; Get parameter + flags
               47    13    2198  5806          BEQL    70$                               ; Br if end of verify table
51   59   0C    00    EF    219A  5807          EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R1     ; Get type field
          51   50    B1    219F  5808          CMPW    R0,R1                             ; Parameters match?
               16    13    21A2  5809          BEQL    30$                               ; Br if yes
               87    B5    21A4  5810          TSTW    (R7)+                             ; Skip offset word
                     21A6  5811          SKIP    PRM_V_MIN,R9,R7                   ; Skip minimum value
                     21AC  5812          SKIP    PRM_V_MAX,R9,R7                   ; Skip maximum value
                     21B2  5813          SKIP    PRM_V_INVALID,R9,R7               ; Skip invalid flags
               DB    11    21B8  5814          BRB     20$                               ; Try next parameter
                     21BA  5815 ;
                     21BA  5816 ; Match found - nullify if same value & check min,max,valid,invalid
                     21BA  5817 ;
          51   87    B0    21BA  5818 30$:    MOVW    (R7)+,R1                          ; Get offset + width
52   51   02    0E    EF    21BD  5819          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R2   ; Get width only
51   51   0E    00    EF    21C2  5820          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1   ; Get offset only
```

```
                                ;
                                ; Byte, word, longword value in structure
                                ;
51    54    C0  21C7  5821          ADDL    R4,R1                           ; Calculate address of datum
                21CA  5822          CASE    R2,TYPE=B,LIMIT=#1,<-           ; Br to handler
                21CA  5823                  40$,-                           ;   Byte value
                21CA  5824                  50$,-                           ;   Word value
                21CA  5825                  60$>                            ;   Longword value
                21D4  5826 ;
                21D4  5827 ; Byte, word, longword value in structure
                21D4  5828 ;
61    55    90  21D4  5829 40$:     MOVB    R5,(R1)                         ; Store byte value
      08    11  21D7  5830          BRB     70$                             ;
61    55    B0  21D9  5831 50$:     MOVW    R5,(R1)                         ; Store word value
      03    11  21DC  5832          BRB     70$                             ;
61    55    D0  21DE  5833 60$:     MOVL    R5,(R1)                         ; Store longword value
   A8 58    F5  21E1  5834 70$:     SOBGTR  R8,10$                          ; Br if more parameters
                21E4  5835
03E6 8F     BA  21E4  5836 80$:     POPR    #^M<R1,R2,R5,R6,R7,R8,R9>       ; Restore registers
                05  21E8  5837          RSB                             ; Return to caller
```

```
                        21E9  5839          .SBTTL  RETURN_P2,  Return UCB/CDB buffer parameters
                        21E9  5840
                        21E9  5841 ;++
                        21E9  5842 ; RETURN_P2 - Return P2 buffer parameters
                        21E9  5843 ;
                        21E9  5844 ; This routine is called to return the UCB/CDB buffer parameters.
                        21E9  5845 ;
                        21E9  5846 ; Inputs:
                        21E9  5847 ;
                        21E9  5848 ;       R1 = Address of return table (same format as verification table)
                        21E9  5849 ;       R3 = IRP address
                        21E9  5850 ;       R4 = UCB or CDB address for storing
                        21E9  5851 ;       R5 = UCB address
                        21E9  5852 ;
                        21E9  5853 ; Outputs:
                        21E9  5854 ;
                        21E9  5855 ;       R0 = destroyed.
                        21E9  5856 ;       All other registers are preserved.
                        21E9  5857 ;
                        21E9  5858 ;--
                        21E9  5859
                        21E9  5860 RETURN_P2:                                   ; Return P2 buffer parameters
    00E6 8F     BB      21E9  5861          PUSHR   #^M<R1,R2,R5,R6,R7>         ; Save registers
56    2C A3     D0      21ED  5862          MOVL    IRP$L_SVAPTE(R3),R6         ; Get system P2 buffer address
          4B    13      21F1  5863          BEQL    60$                         ; Br if no system buffer
    56    66    D0      21F3  5864          MOVL    P2B_L_POINTER(R6),R6        ; Point to start of output buf
                        21F6  5865 ;
                        21F6  5866 ; Loop to return next parameter
                        21F6  5867 ;
    55    81    B0      21F6  5868 10$:     MOVW    (R1)+,R5                    ; Get parameter + flags
          43    13      21F9  5869          BEQL    60$                         ; Br if end of verify table
57  55  0C    00    EF  21FB  5870          EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R5,R7 ; Get type field
          86    57    B0  2200  5871          MOVW    R7,(R6)+                    ; Return parameter
          57    81    B0  2203  5872          MOVW    (R1)+,R7                    ; Get offset + width
52  57  02    0E    EF  2206  5873          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R7,R2 ; Get width only
57  57  0E    00    EF  220B  5874          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R7,R7 ; Get offset only
          57    54    C0  2210  5875          ADDL    R4,R7                       ; Calculate address of datum
                        2213  5876          CASE    R2,TYPE=B,LIMIT=#1,<-       ; Br to handler
                        2213  5877                  20$,-                       ;   Byte value
                        2213  5878                  30$,-                       ;   Word value
                        2213  5879                  40$>                        ;   Longword value
                        221D  5880 ;
                        221D  5881 ; Byte, word, longword value in structure
                        221D  5882 ;
          86    67    9A  221D  5883 20$:     MOVZBL  (R7),(R6)+                  ; Store byte value
                08    11  2220  5884          BRB     50$                         ;
          86    67    3C  2222  5885 30$:     MOVZWL  (R7),(R6)+                  ; Store word value
                03    11  2225  5886          BRB     50$                         ;
          86    67    D0  2227  5887 40$:     MOVL    (R7),(R6)+                  ; Store longword value
                        222A  5888 50$:     SKIP    PRM_V_MIN,R5,R1             ; Skip minimum value
                        2230  5889          SKIP    PRM_V_MAX,R5,R1             ; Skip maximum value
                        2236  5890          SKIP    PRM_V_INVALID,R5,R1         ; Skip invalid flags
                B8      11  223C  5891          BRB     10$                         ; Try for more parameters
                        223E  5892
    00E6 8F     BA      223E  5893 60$:     POPR    #^M<R1,R2,R5,R6,R7>         ; Restore registers
                05      2242  5894          RSB                                 ; Return to caller
```

```
                                2243  5896          .SBTTL  UNPACK_P2_BUF,  Unpack a P2 parameter from P2 buffer
                                2243  5897
                                2243  5898 ;++
                                2243  5899 ; UNPACK_P2_BUF - Unpack a P2 parameter from P2 buffer
                                2243  5900 ;
                                2243  5901 ; Functional description:
                                2243  5902 ;
                                2243  5903 ; This routine is called to get a P2 parameter from the P2 buffer.
                                2243  5904 ;
                                2243  5905 ; Inputs:
                                2243  5906 ;
                                2243  5907 ;       R1 = Parameter type code
                                2243  5908 ;       R3 = IRP address
                                2243  5909 ;       R5 = UCB address
                                2243  5910 ;
                                2243  5911 ; Outputs:
                                2243  5912 ;
                                2243  5913 ;       R0 = Low bit clear if specified Parameter type code is found in P2
                                2243  5914 ;       R2 = Parameter value if success else destroyed
                                2243  5915 ;
                                2243  5916 ;       All other registers are preserved.
                                2243  5917 ;
                                2243  5918 ;--
                                2243  5919
                                2243  5920 UNPACK_P2_BUF:                                    ; Unpack P2 buffer
    00E0 8F     BB              2243  5921          PUSHR   #^M<R5,R6,R7>                   ; Save registers
56    2C A3     D0              2247  5922          MOVL    IRP$L_SVAPTE(R3),R6             ; Get system P2 buffer address
          1C    13              224B  5923          BEQL    20$                             ; Br if none
57    32 A3     3C              224D  5924          MOVZWL  IRP$W_BCNT(R3),R7               ; Get size of P2 buffer
   57    06     C6              2251  5925          DIVL    #6,R7                           ; Caculate number of parameters
56    0C A6     9E              2254  5926          MOVAB   P2B_T_DATA(R6),R6               ; Point to start of data
   50    01     3C              2258  5927          MOVZWL  S^#SS$_NORMAL,R0                ; Assume success
                                225B  5928 ;
                                225B  5929 ; Loop to check next parameter in P2 buffer
                                225B  5930 ;
55    86     3C              225B  5931 10$:     MOVZWL  (R6)+,R5                        ; Get parameter type from P2
52    86     D0              225E  5932          MOVL    (R6)+,R2                        ; Get parameter value from P2
55    51     B1              2261  5933          CMPW    R1,R5                           ; Parameters match?
          05  13              2264  5934          BEQL    30$                             ; Br if yes
      F2 57  F5              2266  5935          SOBGTR  R7,10$                          ; Br if more parameters
                             2269  5936
      50     D4              2269  5937 20$:     CLRL    R0                              ; Return error
00E0  8F     BA              226B  5938 30$:     POPR    #^M<R5,R6,R7>                   ; Restore registers
             05              226F  5939          RSB                                     ; Return to caller
```

```
                                2270  5941          .SBTTL  POKE_USER,  Deliver attention AST
                                2270  5942
                                2270  5943 ;++
                                2270  5944 ; POKE_USER, Deliver attention AST
                                2270  5945 ;
                                2270  5946 ; Functional description:
                                2270  5947 ;
                                2270  5948 ;       This routine is used to deliver an attention AST if one has been
                                2270  5949 ;       requested.
                                2270  5950 ;
                                2270  5951 ; Inputs:
                                2270  5952 ;
                                2270  5953 ;       R5 = UCB address
                                2270  5954 ;       R9 = CDB address
                                2270  5955 ;
                                2270  5956 ; Outputs:
                                2270  5957 ;
                                2270  5958 ;       R0 = Low bit clear only if user is not notified
                                2270  5959 ;       R1-R4 = destroyed.
                                2270  5960 ;
                                2270  5961 ;--
                                2270  5962
                                2270  5963 POKE_USER:                                    ; Poke user process
                                2270  5964          DSBINT  UCB$B_FIPL(R5)              ; Sync access to CDB
                          7E    D4  2277  5965          CLRL    -(SP)                       ; Assume failure
                    51    04 A9   9E  2279  5966          MOVAB   CDB_L_AST(R9),R1            ; Get AST listhead
                          61    D5  227D  5967          TSTL    (R1)                        ; Empty?
                          20    13  227F  5968          BEQL    30$                         ; Branch if yes
                          6E    D6  2281  5969          INCL    (SP)                        ; Indicate success
                          54    DD  2283  5970          PUSHL   R4                          ; Save R4
                    54    51    D0  2285  5971          MOVL    R1,R4                       ; Copy listhead address
                    51    61    D0  2288  5972 10$:     MOVL    (R1),R1                     ; Address a block
                          0B    13  228B  5973          BEQL    20$                         ; Branch if done
                       44 A5    D0  228D  5974          MOVL    UCB$L_DEVDEPEND(R5),-       ; Change parameter
                       1C A1        2290  5975                  ACB$L_KAST+4(R1)            ;    return status
                          69    C8  2292  5976          BISL    CDB_L_DEVDEPEND(R9),-       ; OR in circuit status
                       1C A1        2294  5977                  ACB$L_KAST+4(R1)            ;
                          F0    11  2296  5978          BRB     10$                         ; Continue thru AST blocks
00000000'GF               16  2298  5979 20$:     JSB     G^COM$DELATTNAST            ; Deliver the AST's
                    54 8ED0   229E  5980          POPL    R4                          ; Restore R4
                              22A1  5981
                    50 8ED0   22A1  5982 30$:     POPL    R0                          ; Return success indicator
                              22A4  5983          ENBINT                              ; Restore IPL
                                05  22A7  5984          RSB                         ; Return to caller
```

```
                     22A8  5986          .SBTTL  CANCEL,  Cancel I/O routine
                     22A8  5987
                     22A8  5988 ;++
                     22A8  5989 ; CANCEL, Cancels an I/O operation in progress
                     22A8  5990 ;
                     22A8  5991 ; Functional description:
                     22A8  5992 ;
                     22A8  5993 ;       This routine cancels all I/O on the tributary.
                     22A8  5994 ;
                     22A8  5995 ; Inputs:
                     22A8  5996 ;
                     22A8  5997 ;       R2 = channel number
                     22A8  5998 ;       R3 = current IRP address
                     22A8  5999 ;       R4 = PCB address
                     22A8  6000 ;       R5 = UCB address
                     22A8  6001 ;       R8 = Cancel reason code (zero is for vanila flavored cancel).
                     22A8  6002 ;
                     22A8  6003 ;       IPL = FIPL
                     22A8  6004 ;
                     22A8  6005 ; Outputs:
                     22A8  6006 ;
                     22A8  6007 ;       R0-R3 are destroyed.
                     22A8  6008 ;
                     22A8  6009 ;--
                     22A8  6010
                     22A8  6011 CANCEL:                                      ; Cancel an I/O operation
                     22A8  6013          PUSHQ   R6                          ; Save registers
          0240   30  22AB  6014          BSBW    XD_BUF_JNX                  ; Get a journal buffer
        09 50   E9  22AE  6015          BLBC    R0,1$                       ; Br if error
    86    0A   90  22B1  6016          MOVB    #JNL_CANC,(R6)+             ; Set journal type = CANCEL
    86    52   B0  22B4  6017          MOVW    R2,(R6)+                    ; Enter channel number
    86    58   B0  22B7  6018          MOVW    R8,(R6)+                    ; Enter reason code
                     22BA  6019 1$:     POPQ    R6                          ; Restore registers
                     22BD  6021
                     22BD  6022          ASSUME  CAN$C_CANCEL EQ 0
                     22BD  6023          ASSUME  CAN$C_DASSGN EQ 1
                     22BD  6024          PUSHQ   R8                          ; Save R8, R9
       5C A5    B5  22C0  6025          TSTW    UCB$W_REFC(R5)              ; Last reference to unit?
          0D    12  22C3  6026          BNEQ    5$                          ; Br if no
                     22C5  6027 ;
                     22C5  6028 ; Last deassign on unit
                     22C5  6029 ;
53     24 A5    D0  22C5  6030          MOVL    UCB$L_CRB(R5),R3            ; Get CRB address
        FA70   30  22C9  6031          BSBW    SHUTDOWN_DEV                ; Shutdown the controller
        0122   30  22CC  6032          BSBW    INIT_UCB                    ; Re-initialize the UCB
        00B8   31  22CF  6033          BRW     20$                         ; And leave
                     22D2  6034
50     60 A4    D0  22D2  6035 5$:     MOVL    PCB$L_PID(R4),R0            ; Get PID
   53    52    D0  22D6  6036          MOVL    R2,R3                       ; Copy channel number
        FC35   30  22D9  6037          BSBW    XLATE_PID_CHAN              ; Translate PID and channel
        59 50   E9  22DC  6038          BLBC    R0,8$                       ; Br if none
          58   D5  22DF  6039          TSTL    R8                          ; Is this a plain cancel?
          57   13  22E1  6040          BEQL    10$                         ; Br if yes
                     22E3  6041 ;
                     22E3  6042 ; Deassign channel request
                     22E3  6043 ;
 0218 C541    D4  22E3  6044          CLRL    UCB$L_XD_CDB_VEC(R5)[R1]    ; Remove CDB address from vector
```

```
                0158 C541   D4   22E8  6045          CLRL    UCB$L_XD_PID_VEC(R5)[R1]          ; Zero PID entry
                01D8 C541   B4   22ED  6046          CLRW    UCB$W_XD_CHAN_VEC(R5)[R1]         ; Zero channel entry
                0138 C541   94   22F2  6047          CLRB    UCB$B_XD_TRIB_VEC(R5)[R1]         ; Reset trib address
                   0134 C5  97   22F7  6048          DECB    UCB$B_XD_TRB_CNT(R5)              ; One less tributary
          06 0C A9    00    E0   22FB  6049          BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),7$   ; Br if trib established
                      FB5D  30   2300  6050          BSBW    CLEAR_CDB                         ; Clean up CDB
                      0084  31   2303  6051          BRW     20$                               ; Pop registers and return
                            2306  6052
                      0085  30   2306  6053 7$:      BSBW    DO_CANCEL                         ; Abort UCB xmit I/O
                      F94C  30   2309  6054          BSBW    SHUTDOWN_TRIB1                    ; Clear all CDB I/O
                            230C  6055 ;
                            230C  6056 ; Turn CDB into IRP to halt/kill trib
                            230C  6057 ;
                50    48 A9   D0   230C  6058          MOVL    CDB_L_PID(R9),R0                  ; Save PID
                51    0E A9   90   2310  6059          MOVB    CDB_B_TRB_ADDR(R9),R1             ; Save trib address
                   52    59   D0   2314  6060          MOVL    R9,R2                             ; Copy pointer to IRP
                      00B6  30   2317  6061          BSBW    BLD_IRP                           ; Build the IRP
                38 A3   50   D0   231A  6062          MOVL    R0,IRP$L_MEDIA(R3)                ; Save PID in IRP
                40 A3   51   90   231E  6063          MOVB    R1,IRP$Q_STATION(R3)              ; Store trib address
                21 A3   0E   90   2322  6064          MOVB    #XD_FC_V_HALTT,IRP$B_XDFUNC(R3)   ; Set to halt tributary
                            2326  6065          DSBINT  UCB$B_DIPL(R5)                    ; Sync access to device
                0094 C5   63 0E   232D  6066          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)          ; Insert at front of input Q
                      EC9B  30   2332  6067          BSBW    LOAD_PORT                         ; Give request to device
                            2335  6068          ENBINT                                    ; Restore IPL
                      50    11   2338  6069 8$:      BRB     20$                               ;
                            233A  6070 ;
                            233A  6071 ; Cancel request
                            233A  6072 ;
          58    0218 C541   DE   233A  6073 10$:     MOVAL   UCB$L_XD_CDB_VEC(R5)[R1],R8       ; Save address of trib address
           45 0C A9    00   E1   2340  6074          BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),20$  ; Br if trib not established
            51   00C4 8F   3C   2345  6075          MOVZWL  #IRP$C_LENGTH,R1                  ; Set size of IRP
                            234A  6076 ;
                            234A  6077 ; NOTE - We must use EXE$ALONONPAGED because the EXE$ALLOCIRP resets the
                            234A  6078 ;        IPL to ASTDEL.
                            234A  6079 ;
             00000000'GF    16   234A  6080          JSB     G^EXE$ALONONPAGED                 ; Allocate an IRP
                   09 50    E8   2350  6081          BLBS    R0,15$                            ; Br if success
           50    0124 8F    3C   2353  6082          MOVZWL  #SS$_INSFMEM,R0                   ; Else, return error
                            2358  6083          SETIPL  #0                                ; Enable interrupts
                      04    235B  6084          RET                                       ; RETURN to caller
                            235C  6085
                      0071  30   235C  6086 15$:     BSBW    BLD_IRP                           ; Build the IRP
                44 A3   58   D0   235F  6087          MOVL    R8,IRP$L_CDB(R3)                  ; Save address of CDB address
          0C A3    FAE0 CF   9E   2363  6088          MOVAB   W^RETURN_IRP,IRP$L_PID(R3)        ; Set return address in IRP
            40 A3    0E A9   90   2369  6089          MOVB    CDB_B_TRB_ADDR(R9),IRP$Q_STATION(R3) ; Store trib address
                21 A3   10   90   236E  6090          MOVB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R3)  ; Set function request
                            2372  6091          DSBINT  UCB$B_DIPL(R5)                    ; Sync access to device
                0094 C5   63 0E   2379  6092          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)          ; Insert at front of input Q
                  0C A9   08   AA   237E  6093          BICW    #CD_TS_M_BUFRET,CDB_W_STS(R9)     ; Clear BUFRET flag
                      EC4B  30   2382  6094          BSBW    LOAD_PORT                         ; Give request to device
                            2385  6095          ENBINT                                    ; Restore IPL
                      04    10   2388  6096          BSBB    DO_CANCEL                         ; Abort all IRPs
                            238A  6097 20$:     POPQ    R8                                ; Restore R8, R9
                      05    238D  6098          RSB                                       ; Return to caller
                            238E  6099
                            238E  6100 DO_CANCEL:                                         ; Do the cancel on waiting I/O
                            238E  6101                                                    ; ..pending I/O is canceled
```

```
                     238E  6102                                                ; ..when trib is halted.
 52   009C C5   DE   238E  6103          MOVAL   UCB$Q_XD_XMT_REQ(R5),R2       ; Get address of xmit request Q
            0A   10   2393  6104          BSBB    CHECK_PKT                     ; Run down the queue
   52    18 A9   DE   2395  6105          MOVAL   CDB_Q_RCV_REQ(R9),R2          ; Get address of recv IRP Q
            04   10   2399  6106          BSBB    CHECK_PKT                     ; Run down the queue
   52    10 A9   DE   239B  6107          MOVAL   CDB_Q_XMT_REQ(R9),R2          ; Get address of xmit IRP Q
                     239F  6108                                                ; Run down the queue
                     239F  6109 CHECK_PKT:
                     239F  6110 ;
                     239F  6111 ; NOTE: The xmit request queue on the UCB must only be accessed at FIPL
                     239F  6112 ;
      51    52   D0   239F  6113          MOVL    R2,R1                         ; Copy listhead address
                     23A2  6114          ASSUME  IRP$L_IOQFL EQ 0
      51    61   D0   23A2  6115 10$:     MOVL    (R1),R1                       ; Get next in list
      51    52   D1   23A5  6116          CMPL    R2,R1                         ; End of queue?
            25   13   23A8  6117          BEQL    20$                           ; Br ir yes
      0C    A1   D5   23AA  6118          TSTL    IRP$L_PID(R1)                 ; Is this an Internal IRP?
            0A   14   23AD  6119          BGTR    15$                           ; Br if NO - PID must match
48 A9   02A0 C5   D1   23AF  6120          CMPL    UCB$L_XD_PID(R5),CDB_L_PID(R9) ; Else, ABORT all the Starter's
            EB   12   23B5  6121          BNEQ    10$                           ;  IRPs - but only on a match
            07   11   23B7  6122          BRB     17$                           ; Check the CHAN as well
 48 A9    0C   A1   D1   23B9  6123 15$:     CMPL    IRP$L_PID(R1),CDB_L_PID(R9)   ; Do the PIDs match?
            E2   12   23BE  6124          BNEQ    10$                           ; Br if no - try next packet
 4C A9    28   A1   B1   23C0  6125 17$:     CMPW    IRP$W_CHAN(R1),CDB_W_CHAN(R9) ; Do the Channels match?
            DB   12   23C5  6126          BNEQ    10$                           ; Br if no - try next packet
                     23C7  6127          ASSUME  IRP$L_IOQFL EQ 0
      53    61   0F   23C7  6128          REMQUE  (R1),R3                       ; Remove packet from queue
         F800   30   23CA  6129          BSBW    ABORT_PKT                     ; Abort the IRP
            D0   11   23CD  6130          BRB     CHECK_PKT                     ; Check entire queue
                 05   23CF  6131 20$:     RSB                                   ; Return
                     23D0  6132
                     23D0  6133 ;++
                     23D0  6134 ; BLD_IRP - Build an IRP
                     23D0  6135 ;
                     23D0  6136 ; Inputs:
                     23D0  6137 ;
                     23D0  6138 ;       R2 = address of IRP
                     23D0  6139 ;       R5 = UCB address
                     23D0  6140 ;
                     23D0  6141 ; Outputs:
                     23D0  6142 ;
                     23D0  6143 ;       R2 = Address of IRP$Q_STATION in IRP
                     23D0  6144 ;       R3 = Original address of IRP
                     23D0  6145 ;--
                     23D0  6146
                     23D0  6147 BLD_IRP:                                       ; Build an IRP
      53    82   7E   23D0  6148          MOVAQ   (R2)+,R3                      ; Save IRP address, skip to size fie
                     23D3  6149          ASSUME  IRP$W_SIZE EQ 8
                     23D3  6150          ASSUME  IRP$B_TYPE EQ IRP$W_SIZE+2
                     23D3  6151          ASSUME  IRP$B_RMOD EQ IRP$B_TYPE+1
82   000A0060 8F   D0   23D3  6152          MOVL    #<<DYN$C_IRP@16>!CDB_C_LENGTH>,(R2)+ ; Make it look like an IRP
                     23DA  6153          ASSUME  IRP$L_PID EQ IRP$B_RMOD+1
   82    FA72 CF   9E   23DA  6154          MOVAB   W^RETURN_CDB,(R2)+            ; Store return address from IOPOST
                     23DF  6155          ASSUME  IRP$L_AST EQ IRP$L_PID+4
                     23DF  6156          ASSUME  IRP$L_ASTPRM EQ IRP$L_AST+4
            82   7C   23DF  6157          CLRQ    (R2)+                         ; Clear AST, ASTPRM
                     23E1  6158          ASSUME  IRP$L_WIND EQ IRP$L_ASTPRM+4
```

```
                82   D4  23E1  6159          CLRL    (R2)+                                   ; Clear WIND
                     23E3  6160          ASSUME  IRP$L_UCB EQ IRP$L_WIND+4
            82   55   D0  23E3  6161          MOVL    R5,(R2)+                                ; Store UCB address
                     23E6  6162          ASSUME  IRP$W_FUNC EQ IRP$L_UCB+4
                     23E6  6163          ASSUME  IRP$B_EFN EQ IRP$W_FUNC+2
                     23E6  6164          ASSUME  IRP$B_PRI EQ IRP$B_EFN+1
                     23E6  6165          ASSUME  IRP$L_IOSB EQ IRP$B_PRI+1
                82   7C  23E6  6166          CLRQ    (R2)+                                   ; Clear FUNC, EFN, PRI, IOSB
                     23E8  6167          ASSUME  IRP$W_CHAN EQ IRP$L_IOSB+4
                     23E8  6168          ASSUME  IRP$W_STS EQ IRP$W_CHAN+2
                     23E8  6169          ASSUME  IRP$L_SVAPTE EQ IRP$W_STS+2
                82   7C  23E8  6170          CLRQ    (R2)+                                   ; Clear CHAN, STS, SVAPTE
                     23EA  6171          ASSUME  IRP$W_BOFF EQ IRP$L_SVAPTE+4
                     23EA  6172          ASSUME  IRP$W_BCNT EQ IRP$W_BOFF+2
                     23EA  6173          ASSUME  IRP$L_MEDIA EQ IRP$Q_BCNT+6             ; BCNT is really a longword
                82   7C  23EA  6174          CLRQ    (R2)+                                   ; Clear BOFF, BCNT
                82   7C  23EC  6175          CLRQ    (R2)+                                   ; Clear MEDIA
                     23EE  6176          ASSUME  IRP$Q_STATION EQ IRP$L_MEDIA+8
                82   7C  23EE  6177          CLRQ    (R2)+                                   ; Clear STATION
                     05  23F0  6178          RSB                                             ; Return to caller
```

```
                        23F1  6180          .SBTTL  INIT_UCB,  Initialize UCB parameters
                        23F1  6181
                        23F1  6182 ;++
                        23F1  6183 ; INIT_UCB - Initialize UCB parameters
                        23F1  6184 ;
                        23F1  6185 ; Functional description:
                        23F1  6186 ;
                        23F1  6187 ; This routine is called to reset the UCB default parameters when
                        23F1  6188 ; the last channel is deassigned or wnen the unit is initialized.
                        23F1  6189 ;
                        23F1  6190 ; Inputs:
                        23F1  6191 ;
                        23F1  6192 ;       R5 = UCB address
                        23F1  6193 ;
                        23F1  6194 ;       IRP = FIPL or higher
                        23F1  6195 ;
                        23F1  6196 ; Outputs:
                        23F1  6197 ;
                        23F1  6198 ;       R0-R2 are destroyed.
                        23F1  6199 ;
                        23F1  6200 ;--
                        23F1  6201
                        23F1  6202 INIT_UCB:                                    ; Initialize the UCB
                        23F1  6203          ASSUME  UCB$B_XD_OUTTIM EQ UCB$B_XD_INTIM+1
    02A5 C5     B4      23F1  6204          CLRW    UCB$B_XD_INTIM(R5)          ; Stop all timers
                        23F5  6205          ASSUME  UCB$B_XD_DUP EQ UCB$B_XD_PRO+1
                        23F5  6206          ASSUME  UCB$B_XD_CON EQ UCB$B_XD_DUP+1
                        23F5  6207          ASSUME  UCB$B_XD_BFN EQ UCB$B_XD_CON+1
    02A8 C5     D4      23F5  6208          CLRL    UCB$B_XD_PRO(R5)            ; Reset all line characteristics
42  A5  0200 8F    B0  23F9  6209          MOVW    #XD_DEF_BUFSIZ,UCB$W_DEVBUFSIZ(R5) ; Set default buffer size
          04    90      23FF  6210          MOVB    #MAX_XMT_TRB,-              ; Set maximum number of xmits
    0135 C5             2401  6211                  UCB$B_XD_XMT_TRB(R5)        ; ..per tributary
    50    0F    3C      2404  6212          MOVZWL  #DEF_LINE_PARAMSZ,R0        ; Set size of defaults in bytes
51  DD2D CF     9E      2407  6213          MOVAB   DEF_LINE_PARAM,R1           ; Set address of defaults
52  02A7 C5     9E      240C  6214          MOVAB   UCB$B_XD_SETPRM(R5),R2      ; Set address of parameters
    82    81    90      2411  6215 10$:     MOVB    (R1)+,(R2)+                 ; Set next default
      FA  50    F5      2414  6216          SOBGTR  R0,10$                      ; Loop on all parameters
                05      2417  6217          RSB
```

```
                     2418  6219          .SBTTL  Timer setup routines
                     2418  6220
                     2418  6221 ;++
                     2418  6222 ; Timer setup routines
                     2418  6223 ;
                     2418  6224 ; Functional description:
                     2418  6225 ;
                     2418  6226 ; These routines set/reset the input and output wait timers.
                     2418  6227 ;
                     2418  6228 ; Inputs:
                     2418  6229 ;
                     2418  6230 ;       R5 = UCB address
                     2418  6231 ;
                     2418  6232 ; Outputs:
                     2418  6233 ;
                     2418  6234 ;       R0,R1 are destroyed.
                     2418  6235 ;
                     2418  6236 ;--
                     2418  6237
                     2418  6238 SET_INTIM:                                    ; Start input timer
51   0094 C5   9E   2418  6239          MOVAB   UCB$Q_XD_INPUTQ(R5),R1        ; Get address of input queue
     51   61   D1   241D  6240          CMPL    (R1),R1                       ; Anything on queue?
          01   12   2420  6241          BNEQ    20$                           ; Br if yes - reset timer
               05   2422  6242 10$:     RSB                                   ; Else, exit
                    2423  6243
     02A5 C5   95   2423  6244 20$:     TSTB    UCB$B_XD_INTIM(R5)            ; Timer already going?
          F9   12   2427  6245          BNEQ    10$                           ; Br if yes - exit
02A5 C5   03   90   2429  6246          MOVB    #3,UCB$B_XD_INTIM(R5)         ; Reset timeout cell
          41   11   242E  6247          BRB     START_TIMER                   ; Start timer if needed
                    2430  6248
                    2430  6249 SET_OUTTIM:                                   ; Start output timer
                    2430  6250          DSBINT  UCB$B_DIPL(R5)                ; Disable device interrupts
          02A6 C5   94   2437  6251          CLRB    UCB$B_XD_OUTTIM(R5)       ; Clear output timer
     51   00AC C5   9E   243B  6252          MOVAB   UCB$Q_XD_INFOUT(R5),R1    ; Get address of UCB output queue
          51   61   D1   2440  6253          CMPL    (R1),R1                   ; Anything on queue?
               24   12   2443  6254          BNEQ    30$                       ; Br if yes - reset timer
     50   DD08 CF   3C   2445  6255          MOVZWL  MAX_W_TRIB,R0             ; Get maximum number of CDBs
               50   D7   244A  6256          DECL    R0                        ;   Minus 1
51   0218 C540   D0   244C  6257 10$:     MOVL    UCB$L_XD_CDB_VEC(R5)[R0],R1   ; Get address of next CDB
               0E   13   2452  6258          BEQL    20$                       ; Br if none
09 0C A1   00   E1   2454  6259          BBC     #CD_TS_V_ESTAB,CDB_W_STS(R1),20$ ; Br if trib not established
     51   30 A1   9E   2459  6260          MOVAB   CDB_Q_INFOUT(R1),R1       ; Get address of output queue
          51   61   D1   245D  6261          CMPL    (R1),R1                   ; Anything on queue?
               07   12   2460  6262          BNEQ    30$                       ; Br if yes - reset timer
          E7 50   F4   2462  6263 20$:     SOBGEQ  R0,10$                    ; Loop on all CDBs
                    2465  6264          ENBINT                            ; Restore IPL
               05   2468  6265          RSB                               ; Else, exit
                    2469  6266
                    2469  6267 30$:     ENBINT                            ; Restore IPL
     02A6 C5   0A   90   246C  6268          MOVB    #10,UCB$B_XD_OUTTIM(R5)   ; Reset timeout cell
                    2471  6269          ;BRB    START_TIMER               ; Start timer
                    2471  6270
                    2471  6271 START_TIMER:                                 ; Startup timer
     0D 64 A5   00   E0   2471  6272          BBS     #UCB$V_TIM,UCB$W_STS(R5),10$ ; Br if timer going
6C A5   00000000'GF   02   C1   2476  6273          ADDL3   #2,G^EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ; Set 2 second timer
          64 A5   03   A8   247F  6274          BISW    #UCB$M_TIM!UCB$M_INT,UCB$W_STS(R5) ; Enable timer
               05   2483  6275 10$:     RSB
```

```
                        2484  6277          .SBTTL  TIMEOUT,  Timeout Routine
                        2484  6278
                        2484  6279 ;++
                        2484  6280 ; TIMEOUT - Timeout Routine
                        2484  6281 ;
                        2484  6282 ; Functional description:
                        2484  6283 ;
                        2484  6284 ; This routine is entered on tributary timeout.  The action is to set
                        2484  6285 ; the error status and branch to the error routine.
                        2484  6286 ;
                        2484  6287 ; Inputs:
                        2484  6288 ;
                        2484  6289 ;       R5 = UCB address
                        2484  6290 ;
                        2484  6291 ; Outputs:
                        2484  6292 ;
                        2484  6293 ;       R3,R4 are destroyed.
                        2484  6294 ;       R5 is preserved.
                        2484  6295 ;
                        2484  6296 ;--
                        2484  6297
                        2484  6298 TIMEOUT:
   64 A5   03   AA      2484  6299          BICW    #UCB$M_TIM!UCB$M_INT,UCB$W_STS(R5) ; Disable timer
3F 68 A5   00   E1      2488  6300          BBC     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),40$ ; Br if not inited
 53   01   15   78      248D  6301          ASHL    #XD_BSEL2_V_ERR+16,#1,R3        ; Set error indicator
 54   01   10   78      2491  6302          ASHL    #16,#1,R4                       ; Assume powerfail
2F 64 A5   05   E0      2495  6303          BBS     #UCB$V_POWER,UCB$W_STS(R5),30$  ; Br if powerfail
      02A5 C5   95      249A  6304          TSTB    UCB$B_XD_INTIM(R5)              ; Input timer going?
           09   13      249E  6305          BEQL    10$                             ; Br if not
      02A5 C5   97      24A0  6306          DECB    UCB$B_XD_INTIM(R5)              ; Input wait timeout?
           0F   13      24A4  6307          BEQL    20$                             ; Br if yes - error
           C8   AF      9F    24A6  6308          PUSHAB  B^START_TIMER                   ; Else, remember to restart timer
      02A6 C5   95      24A9  6309 10$:     TSTB    UCB$B_XD_OUTTIM(R5)             ; Output timer going?
           1D   13      24AD  6310          BEQL    40$                             ; Br if not - exit
      02A6 C5   97      24AF  6311          DECB    UCB$B_XD_OUTTIM(R5)             ; Output wait timeout?
           BC   12      24B3  6312          BNEQ    START_TIMER                     ; Br it not - restart timer
    54   24 A5   D0      24B5  6313 20$:     MOVL    UCB$L_CRB(R5),R4                ; Get CRB address
                         24B9  6314          ASSUME  IDB$L_CSR EQ 0
    54   2C B4   D0      24B9  6315          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4    ; Get CSR address
01 A4   40 8F   90      24BD  6316          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)      ; Stop the device
           54   D4      24C2  6317          CLRL    R4                              ; Indicate timeout
                        24C4  6318          SETBIT  #XM$V_STS_TIMO,UCB$L_DEVDEPEND(R5) ; Set error status
         F3D7   30      24C9  6319 30$:     BSBW    SCHED_FORK                      ; Schedule a fork process
                05      24CC  6320 40$:     RSB                                     ; Return to caller
```

```
                        24CD  6323          .SBTTL  XD_FILL_JNX,  Fill a journal record
                        24CD  6324          .SBTTL  XD_BUF_JNX,  Get a journal record
                        24CD  6325
                        24CD  6326 ;++
                        24CD  6327 ; XD_FILL_JNX - Fill a journal record
                        24CD  6328 ; XD_BUF_JNX - Get a journal record
                        24CD  6329 ;
                        24CD  6330 ; Functional description:
                        24CD  6331 ;
                        24CD  6332 ; This routine finds a journal record and stores the IRP information
                        24CD  6333 ; into the journal record.
                        24CD  6334 ;
                        24CD  6335 ; Inputs:
                        24CD  6336 ;
                        24CD  6337 ;       R0 = Journal record type
                        24CD  6338 ;       R3 = IRP address (XD_FILL_JNX only)
                        24CD  6339 ;       R5 = UCB address
                        24CD  6340 ;
                        24CD  6341 ; Outputs:
                        24CD  6342 ;       R0 = success indicator
                        24CD  6343 ;       R6,R7 are destroyed.
                        24CD  6344 ;
                        24CD  6345 ;--
              00000020  24CD  6346          JNL_REC_SIZ = 32
                        24CD  6347
                        24CD  6348 XD_FILL_JNX::                                ; Fill the journal record
                50   DD  24CD  6349          PUSHL   R0                      ; Save record type
                1D   10  24CF  6350          BSBB    XD_BUF_JNX              ; Get a journal record
             16 50   E9  24D1  6351          BLBC    R0,90$                  ; Br if error
                50 8ED0  24D4  6352          POPL    R0                      ; Restore record type
             86   50   90  24D7  6353          MOVB    R0,(R6)+                ; Enter record type
          86   28 A3   B0  24DA  6354          MOVW    IRP$W_CHAN(R3),(R6)+    ; Enter CHAN
          86   20 A3   B0  24DE  6355          MOVW    IRP$W_FUNC(R3),(R6)+    ; Enter FUNC
          86   40 A3   7D  24E2  6356          MOVQ    IRP$Q_STATION(R3),(R6)+ ; Enter STATION
             50   01   9A  24E6  6357          MOVZBL  #1,R0                   ; Return with success
                       05  24E9  6358          RSB
                           24EA  6359
             5E   04   C0  24EA  6360 90$:     ADDL    #4,SP                   ; Pop stack
                       05  24ED  6361          RSB                             ; Return
                           24EE  6362
                           24EE  6363 XD_BUF_JNX::                             ; Get a journal record
                           24EE  6364          DSBINT  UCB$B_DIPL(R5)          ; Sync access to buffers
          57   0090 C5   D0  24F5  6365          MOVL    UCB$L_XD_JNLBUF(R5),R7  ; Get the journal buffer
                    06   13  24FA  6366          BEQL    100$                    ; Br if none
             06 A7   20   B1  24FC  6367          CMPW    #JNL_REC_SIZ,6(R7)      ; Is there enough space?
                    06   1F  2500  6368          BLSSU   200$                    ; Br if no
                           2502  6369 100$:     ENBINT                          ; Re-enable interrupts
                50   D4  2505  6370          CLRL    R0                      ; Return error
                       05  2507  6371          RSB
                           2508  6372
             06 A7   20   A2  2508  6373 200$:     SUBW    #JNL_REC_SIZ,6(R7)      ; Acquire space to be used
                56   67   D0  250C  6374          MOVL    (R7),R6                 ; Get output pointer
                67   20   C0  250F  6375          ADDL    #JNL_REC_SIZ,(R7)       ; Adjust output pointer
                           2512  6376          ENBINT                          ; Re-enable interrupts
                57   56   D0  2515  6377          MOVL    R6,R7                   ; Copy buffer pointer
       86 00000000'GF   7D  2518  6378          MOVQ    G^EXE$GQ_SYSTIME,(R6)+  ; Enter timestamp
                50   01   9A  251F  6379          MOVZBL  #1,R0                   ; Return with success
```

```
 05  2522  6380          RSB                                 ; Return to caller
     2523  6381
     2523  6383
     2523  6384
     2523  6385
     2523  6386 XD_END:                                      ; Last location in driver
     2523  6387          .END
```

```
$$$                             = 00000020 R      02
$$$OFF                          = 0000005C
$$$TYP                          = 00000406
$$$WID                          = 00002000
$$BASE                          = 00000001
$$DISPL                         = 00000008
$$GENSW                         = 00000001
$$HIGH                          = 00000007
$$LIMIT                         = 00000006
$$LOW                           = 00000001
$$MNSW                          = 00000001
$$MXSW                          = 00000001
$$OP                            = 00000002
ABORTIO                           000002D5 R      03
ABORT_PKT                         00001BCD R      03
ACB$L_KAST                      = 00000018
ACCESS                            00000AF4 R      03
ADDRCVLIST                        00000EE0 R      03
ALLOC_P2BUF                       00000B0D R      03
ALT_START                         000004AE R      03
AT$_UBA                         = 00000001
BLD_IRP                           000023D0 R      03
BSEL0                             00000000
BSEL1                             00000001
BSEL2                             00000002
BSEL3                             00000003
BSEL4                             00000004
BSEL5                             00000005
BSEL6                             00000006
BSEL7                             00000007
BUG$_NOBUFPCKT                    ********   X    03
BUG$_UNSUPRTCPU                   ********   X    03
CAN$C_CANCEL                    = 00000000
CAN$C_DASSGN                    = 00000001
CANCEL                            000022A8 R      03
CDB_B_MAX_MSGS                    0000005B   G
CDB_B_MRB                         0000004E   G
CDB_B_MST                         0000004F   G
CDB_B_POL                         0000000B   G
CDB_B_POL_QACT                    00000052   G
CDB_B_POL_QDYI                    00000056   G
CDB_B_POL_QIN                     00000054   G
CDB_B_POL_RACT                    00000053   G
CDB_B_POL_RDYI                    00000057   G
CDB_B_POL_RIN                     00000055   G
CDB_B_SETPRM                      0000004E   G
CDB_B_TO_DEAD                     0000005A   G
CDB_B_TO_DYING                    00000059   G
CDB_B_TO_INACT                    00000058   G
CDB_B_TRB_ADDR                    0000000E   G
CDB_B_TYPE                        0000000A   G
CDB_B_XMT_CNT                     0000000F   G
CDB_C_CNTS                      = 00000010
CDB_C_LENGTH                      00000060   G
CDB_C_QUEUES                    = 00000005
CDB_C_SETPRM                    = 00000012
CDB_L_AST                         00000004   G
CDB_L_BRC                         00000038   G
CDB_L_BSN                         0000003C   G
CDB_L_CNTS                        00000038   G
CDB_L_DEVDEPEND                   00000000   G
CDB_L_DMR                         00000040   G
CDB_L_DMS                         00000044   G
CDB_L_PID                         00000048   G
CDB_Q_INFOUT                      00000030   G
CDB_Q_QUEUES                      00000010   G
CDB_Q_RCV_MSG                     00000020   G
CDB_Q_RCV_REQ                     00000018   G
CDB_Q_XMT_PND                     00000028   G
CDB_Q_XMT_REQ                     00000010   G
CDB_W_BAB_TIMER                   0000005E   G
CDB_W_CHAN                        0000004C   G
CDB_W_DEL_TIMER                   00000050   G
CDB_W_POLLPRM                     00000050   G
CDB_W_SEL_TIMER                   0000005C   G
CDB_W_SIZE                        00000008   G
CDB_W_STS                         0000000C   G
CD_TS_M_BUFQUO                  = 00000010   G
CD_TS_M_BUFRET                  = 00000008   G
CD_TS_M_CANCEL                  = 00000004   G
CD_TS_M_ESTAB                   = 00000001   G
CD_TS_M_MOP                     = 00000002   G
CD_TS_M_START                   = 00000020   G
CD_TS_V_BUFQUO                  = 00000004   G
CD_TS_V_BUFRET                  = 00000003   G
CD_TS_V_CANCEL                  = 00000002   G
CD_TS_V_ESTAB                   = 00000000   G
CD_TS_V_MOP                     = 00000001   G
CD_TS_V_START                   = 00000005   G
CHECK_BOFS                        00000ACE R      03
CHECK_P1                          00000AFA R      03
CHECK_P2                          00000AD0 R      03
CHECK_PKT                         0000239F R      03
CHG_CDB                           00002039 R      03
CHG_UCB                           00001F66 R      03
CLEAR_CDB                         00001E60 R      03
COM$DELATTNAST                    ********   X    03
COM$DRVDEALMEM                    ********   X    03
COM$FLUSHATTNS                    ********   X    03
COM$POST                          ********   X    03
COM$SETATTNAST                    ********   X    03
CRB$L_INTD                      = 00000024
CRB$L_INTD2                     = 00000048
CXB$B_CODE                      = 0000000B
CXB$B_TYPE                      = 0000000A
CXB$C_HEADER                    = 00000048
CXB$C_TRAILER                   = 00000004
CXB$L_BL                        = 00000004
CXB$L_FL                        = 00000000
CXB$W_LENGTH                    = 0000000C
CXB$W_SIZE                      = 00000008
DC$_SCOM                        = 00000020
DDB$L_DDT                       = 0000000C
DEAD_ERR                          00001799 R      03
```

```
DEF_LINE_PARAM                00000138 RG     03
DEF_LINE_PARAMSZ            = 0000000F
DEF_TRIB_PARAM                00000126 RG     03
DEF_TRIB_PARAMSZ            = 00000012
DEV$M_AVC                   = 00040000
DEV$M_IDV                   = 04000000
DEV$M_NET                   = 00002000
DEV$M_ODV                   = 08000000
DEV$M_SHR                   = 00010000
DEV_ACTIVE                    00000E5B R      03
DEV_INACT                     000014A2 R      03
DEV_INACT1                    00001285 R      03
DEV_INACT_SP                  000014A0 R      03
DO_CANCEL                     0000238E R      03
DPT$C_LENGTH                = 00000038
DPT$C_VERSION               = 00000004
DPT$INITAB                    00000038 R      02
DPT$REINITAB                  0000004F R      02
DPT$TAB                       00000000 R      02
DRV_DONE                      00000BC6 R      03
DSC$A_POINTER               = 00000004
DT$_DAP11                   = 00000009
DT$_DMV11                   = 00000017
DYN$C_BUFIO                 = 00000013
DYN$C_CRB                   = 00000005
DYN$C_CXB                   = 0000001B
DYN$C_DDB                   = 00000006
DYN$C_DPT                   = 0000001E
DYN$C_FRK                   = 00000008
DYN$C_IRP                   = 0000000A
DYN$C_NET                   = 00000017
DYN$C_UCB                   = 00000010
EXE$ABORTIO                   ********    X 03
EXE$ALLOCBUF                  ********    X 03
EXE$ALONONPAGED               ********    X 03
EXE$BUFFRQUOTA                ********    X 03
EXE$BUFQUOPRC                 ********    X 03
EXE$FINISHIO                  ********    X 03
EXE$FORK                      ********    X 03
EXE$GB_CPUTYPE                ********    X 03
EXE$GL_ABSTIM                 ********    X 03
EXE$GL_TENUSEC                ********    X 03
EXE$GL_UBDELAY                ********    X 03
EXE$GQ_SYSTIME                ********    X 03
EXE$PROBER_DSC                ********    X 03
EXE$QIORETURN                 ********    X 03
EXE$READCHK                   ********    X 03
EXE$WRITECHK                  ********    X 03
FATAL_ERR                     000017AB R      03
FILLRCVLIST                   00000ED9 R      03
FIND_SLOT                     00001E97 R      03
FIND_TRIB                     00001EC4 R      03
FINISH_RCV_IO                 00001B93 R      03
FKB$B_FIPL                  = 0000000B
FKB$B_TYPE                  = 0000000A
FKB$C_LENGTH                = 00000018
FKB$L_FR3                   = 00000010
FORCE_HALT                    00001718 R      03
FORCE_HALT1                   0000171C R      03
FORK_CANCEL                   00001B0F R      03
FORK_COUNTER                  00001A1E R      03
FORK_HALTT                    00001AF2 R      03
FORK_INITC                    0000199B R      03
FORK_KILLT                    00001B24 R      03
FORK_PKT                      00001991 R      03
FORK_POLSS                    00001B2A R      03
FORK_PROC                     000018CC R      03
FORK_RECV                     00001ADA R      03
FORK_SSLOT                    00001A4B R      03
FORK_STOPC                    00001A12 R      03
FORK_STOPT                    000019E2 R      03
FORK_XMIT                     00001A5B R      03
FUNCTAB_LEN                 = 00000040
GET_CDB                       00001F50 R      03
GET_CHAR_BUFS                 0000086A R      03
HALT_TRIBS                    00001C1A R      03
IDB$L_CSR                   = 00000000
IDB$L_UCBLST                = 00000018
INF_RCV                     = 000000FF
INF_XMT                     = 0000007F
INIT_OTHERS                   0000022D R      03
INIT_UCB                      000023F1 R      03
INIT_UV1                      0000019D R      03
INTERR                        00001514 R      03
INTEXIT                       00001507 R      03
IN_CANCEL                     000013FC R      03
IN_CHGC                       00001221 R      03
IN_CHGT                       00001277 R      03
IN_DISP                       000010D7 R      03
IN_HALTT                      00001453 R      03
IN_INITC                      0000116D R      03
IN_INITT                      0000110B R      03
IN_KILLT                      000011E7 R      03
IN_NUMSYNC                    0000125F R      03
IN_POLSTAT                    000012BF R      03
IN_RCGERR                     0000147B R      03
IN_RCTERR                     000013B7 R      03
IN_RDGERR                     00001476 R      03
IN_RDGSS                      0000146B R      03
IN_RDMODEM                    0000130E R      03
IN_RDTERR                     000013B2 R      03
IN_RDTSS                      0000139E R      03
IN_READG                      00001494 R      03
IN_READT                      000013DC R      03
IN_RECV                       0000137E R      03
IN_STOPC                      000011FC R      03
IN_STOPT                      00001188 R      03
IN_WTMODEM                    000013EC R      03
IN_XMIT                       00001326 R      03
IO$V_ATTNAST                = 00000008
IO$V_CLR_COUNT              = 0000000A
IO$V_CTRL                   = 00000009
IO$V_NOW                    = 00000006
IO$V_RD_COUNT               = 00000008
```

```
IO$V_RD_MEM                     =  00000006
IO$V_RD_MODEM                   =  00000007
IO$V_SET_MODEM                  =  0000000A
IO$V_SHUTDOWN                   =  00000007
IO$V_STARTUP                    =  00000006
IO$_READLBLK                    =  00000021
IO$_READPBLK                    =  0000000C
IO$_READVBLK                    =  00000031
IO$_SENSEMODE                   =  00000027
IO$_SETCHAR                     =  0000001A
IO$_SETMODE                     =  00000023
IO$_VIRTUAL                     =  0000003F
IO$_WRITELBLK                   =  00000020
IO$_WRITEPBLK                   =  0000000B
IO$_WRITEVBLK                   =  00000030
IOC$ALOUBAMAP                   ********    X   03
IOC$INITIATE                    ********    X   03
IOC$LOADUBAMAPA                 ********    X   03
IOC$LOADUBAMAPN                 ********    X   03
IOC$MNTVER                      ********    X   03
IOC$RELMAPREG                   ********    X   03
IOC$REQMAPREG                   ********    X   03
IOC$RETURN                      ********    X   03
IOC$WFIKPCH                     ********    X   03
IO_DONE                         00001BD0  R     03
IO_DONE1                        00001BD4  R     03
IRP$B_EFN                       =  00000022
IRP$B_INDEX                        00000041
IRP$B_POLS                         0000003F
IRP$B_PRI                       =  00000023
IRP$B_RMOD                      =  0000000B
IRP$B_TYPE                      =  0000000A
IRP$B_XDFUNC                       00000021
IRP$C_LENGTH                    =  000000C4
IRP$L_AST                       =  00000010
IRP$L_ASTPRM                    =  00000014
IRP$L_CDB                          00000044
IRP$L_DIAGBUF                   =  0000004C
IRP$L_IOQFL                     =  00000000
IRP$L_IOSB                      =  00000024
IRP$L_IOST1                     =  00000038
IRP$L_IOST2                     =  0000003C
IRP$L_MEDIA                     =  00000038
IRP$L_PID                       =  0000000C
IRP$L_SVAPTE                    =  0000002C
IRP$L_UCB                       =  0000001C
IRP$L_WIND                      =  00000018
IRP$Q_STATION                   =  00000040
IRP$V_DIAGBUF                   =  00000007
IRP$V_FUNC                      =  00000001
IRP$W_BCNT                      =  00000032
IRP$W_BOFF                      =  00000030
IRP$W_CHAN                      =  00000028
IRP$W_CSTATUS                      00000042
IRP$W_FUNC                      =  00000020
IRP$W_QUOTA                        00000042
IRP$W_SIZE                      =  00000008
IRP$W_STS                       =  0000002A
JIB$L_BYTCNT                    =  00000020
JIB$L_BYTLM                     =  00000024
JNL_CANC                        =  0000C00A
JNL_CSRIN                       =  00000008
JNL_CSROUT                      =  00000009
JNL_POST                        =  00000005
JNL_QIO                         =  00000003
JNL_QUE                         =  00000004
JNL_RCV                         =  00000002
JNL_RCVA                        =  00000007
JNL_REC_SIZ                     =  00000020
JNL_XMT                         =  00000001
JNL_XMTA                        =  00000006
JNX$$$                          =  00000001
LINE_CNT_BUFSIZ                 =  0000000A
LINE_COUNTER                       00000153  R     03
LINE_PARAM                         00000078  R     03
LINE_PRM_BUFSIZ                 =  00000042
LOAD_PORT                          00000FD0  R     03
LOAD_PORT_AVAIL                    0000101F  R     03
MAINT_ERR                          0000178F  R     03
MASKH                           =  00000080
MASKL                           =  00000000
MAX_BUFSIZ                      =  00003FFF
MAX_BUFSIZ_UV1                  -  00000400
MAX_DMP_TRIB                    =  00000020
MAX_DMV_TRIB                    =  00000010
MAX_RCV                         =  00000007
MAX_RCV_UV1                     =  00000004
MAX_W_BUFFER                       0000014F  R     03
MAX_W_TRIB                         00000151  R     03
MAX_XMT                         =  00000007
MAX_XMT_TRB                     =  00000004
MAX_XMT_UV1                     =  00000002
MMG$GL_SPTBASE                     ********    X   03
NMA$C_CIRPST_ACT                =  00000002
NMA$C_CIRPST_AUT                =  00000001
NMA$C_CIRPST_DED                =  00000005
NMA$C_CIRPST_DIE                =  00000004
NMA$C_CIRPST_INA                =  00000003
NMA$C_CTCIR_BRC                 =  000003E8
NMA$C_CTCIR_BSN                 =  000003E9
NMA$C_CTCIR_DBR                 =  000003F2
NMA$C_CTCIR_DBS                 =  000003F3
NMA$C_CTCIR_DEI                 =  000003FC
NMA$C_CTCIR_DEO                 =  000003FD
NMA$C_CTCIR_LBE                 =  00000411
NMA$C_CTCIR_LRT                 =  00000407
NMA$C_CTCIR_RBE                 =  00000410
NMA$C_CTCIR_RRT                 =  00000406
NMA$C_CTCIR_SIE                 =  0000041A
NMA$C_CTCIR_SLT                 =  0000041B
NMA$C_CTLIN_LPE                 =  0000044D
NMA$C_CTLIN_RPE                 =  0000044C
NMA$C_DPX_FDL                   =  00000000
NMA$C_DPX_HAL                   =  00000001
```

```
NMA$C_LINCN_LOO = 00000001
NMA$C_LINCN_NOR = 00000000
NMA$C_LINPR_CON = 00000001
NMA$C_LINPR_DMC = 00000004
NMA$C_LINPR_POI = 00000000
NMA$C_LINPR_TRI = 00000002
NMA$C_PCCI_ACB = 0000047E
NMA$C_PCCI_ACI = 0000047F
NMA$C_PCCI_BBT = 00000475
NMA$C_PCCI_DTH = 00000486
NMA$C_PCCI_DYB = 00000483
NMA$C_PCCI_DYI = 00000484
NMA$C_PCCI_DYT = 00000485
NMA$C_PCCI_IAB = 00000480
NMA$C_PCCI_IAI = 00000481
NMA$C_PCCI_IAT = 00000482
NMA$C_PCCI_MRB = 00000479
NMA$C_PCCI_MST = 00000AFA
NMA$C_PCCI_MTR = 0000047A
NMA$C_PCCI_PLS = 000003F3
NMA$C_PCCI_POL = 000003F2
NMA$C_PCCI_RTT = 00000477
NMA$C_PCCI_TRI = 00000474
NMA$C_PCCI_TRT = 00000476
NMA$C_PCLI_BFN = 00000451
NMA$C_PCLI_BUS = 00000AF1
NMA$C_PCLI_CON = 00000456
NMA$C_PCLI_DDT = 0000047F
NMA$C_PCLI_DLT = 00000480
NMA$C_PCLI_DUP = 00000457
NMA$C_PCLI_NMS = 00000AFA
NMA$C_PCLI_PRO = 00000458
NMA$C_PCLI_RTT = 00000461
NMA$C_PCLI_SLT = 0000047E
NMA$C_PCLI_SRT = 00000481
NMA$C_STATE_OFF = 00000001
NMA$C_STATE_ON = 00000000
NMA$M_CNT_COU = 00008000
NMA$M_CNT_MAP = 00001000
NMA$M_CNT_TYP = 00000FFF
NMA$V_CNT_MAP = 0000000C
NMA$V_CNT_WID = 0000000D
NOOP 0000177A R 03
OFF_M_VALUE = 00003FFF
OFF_S_VALUE = 0000000E
OFF_S_WIDTH = 00000002
OFF_V_VALUE = 00000000
OFF_V_WIDTH = 0000000E
OUT_CTL 000016E4 R 03
OUT_INFO 0000168F R 03
OUT_RCV 00001590 R 03
OUT_RCV_ERR 0000166C R 03
OUT_STATUS 000017CA R 03
OUT_XMIT 0000162E R 03
OUT_XMIT_ERR 0000165B R 03
P1 = 00000000
P1_LINE_MASK = 000000FF
P1_TRIB_MASK = 000028FF
P2 = 00000004
P2B_B_SPARE 0000000B
P2B_B_TYPE 0000000A
P2B_C_LENGTH 0000000C
P2B_L_BUFFER 00000004
P2B_L_POINTER 00000000
P2B_T_DATA 0000000C
P2B_W_SIZE 00000008
P3 = 00000008
P4 = 0000000C
P5 = 00000010
PCB$L_JIB = 00000080
PCB$L_PID = 00000060
POKE_USER 00002270 R 03
POLSS_TBL 00000147 R 03
POST_IT 00001C01 R 03
PR$_IPL = 00000012
PR$_SID_TYP730 = 00000003
PR$_SID_TYP750 = 00000002
PR$_SID_TYP780 = 00000001
PR$_SID_TYP790 = 00000004
PR$_SID_TYPUV1 = 00000007
PRM_M_INVALID = 00004000
PRM_M_MAX = 00002000
PRM_M_MIN = 00001000
PRM_M_TYPE = 00000FFF
PRM_S_TYPE = 0000000C
PRM_V_INVALID = 0000000E
PRM_V_MAX = 0000000D
PRM_V_MIN = 0000000C
PRM_V_TYPE = 00000000
PROCEDURE_ERR 00001710 R 03
PTE$S_PFN = 00000015
PTE$V_PFN = 00000000
QUEPKT 00000ABE R 03
RCV_B_BAHI 00000010
RCV_B_BLKTYPE 0000000A
RCV_B_MAPSLOT 0000000B
RCV_FDT 00000418 R 03
RCV_L_BACC 0000000C
RCV_L_LINK 00000000
RCV_START 00000458 R 03
RCV_START_ALT 0000048C R 03
RCV_T_DATA 00000048
RCV_W_BLKSIZE 00000008
RDI_INTRPT 000014CD R 03
RDO_INTRPT 0000151E R 03
REG_DUMP 00001E3A R 03
REQ_ABORT 000014B1 R 03
REQ_COM 000014BE R 03
REQ_COM1 0000121E R 03
REQ_COM_ERR 000014C1 R 03
RETURN_CDB 00001E50 R 03
RETURN_IRP 00001E47 R 03
RETURN_P2 000021E9 R 03
RUNNING_NOTIF 0000179F R 03
```

```
SCH$GL_PCBVEC                   ********   X    03        UCB$B_DEVCLASS                  =  00000040
SCHED_FORK                      000018A3   R    03        UCB$B_DEVTYPE                   =  00000041
SCHED_FORKC                     000018A1   R    03        UCB$B_DIPL                      =  0000005E
SEL0                            00000000                  UCB$B_FIPL                      =  0000000B
SEL10                           00000008                  UCB$B_XD_BFN                       000002AB  G
SEL2                            00000002                  UCB$B_XD_CON                       000002AA  G
SEL4                            00000004                  UCB$B_XD_DUP                       000002A9  G
SEL6                            00000006                  UCB$B_XD_FKB                       000002B6  G
SENSEMODE_CTRL                  000009D7   R    03        UCB$B_XD_INTIM                     000002A5  G
SENSEMODE_FDT                   000008C0   R    03        UCB$B_XD_MODE                      000002A4  G
SENSE_MODEM                     00000AA0   R    03        UCB$B_XD_NMS                       000002A7  G
SETMODE_CTRL                    00000739   R    03        UCB$B_XD_OUTTIM                    000002A6  G
SETMODE_FDT                     0000050A   R    03        UCB$B_XD_PRO                       000002A8  G
SET_INDEX                       00001731   R    03        UCB$B_XD_RCV_MAP                   00000130  G
SET_INTIM                       00002418   R    03        UCB$B_XD_RCV_MAX                   00000132  G
SET_OUTTIM                      00002430   R    03        UCB$B_XD_SETPRM                    000002A7  G
SFORK_EXIT                      0000174B   R    03        UCB$B_XD_TRB_CNT                   00000134  G
SHUTDOWN_DEV                    00001D3C   R    03        UCB$B_XD_TRIB_VEC                  00000138  G
SHUTDOWN_TRIB                   00001C6F   R    03        UCB$B_XD_XMT_MAP                   00000131  G
SHUTDOWN_TRIB1                  00001C58   R    03        UCB$B_XD_XMT_MAX                   00000133  G
SIZ...                      =   00000002                  UCB$B_XD_XMT_TRB                   00000135  G
SOFT_ERR                        00001750   R    03        UCB$C_LENGTH                    =  00000090
SS$_ABORT                   =   0000002C                  UCB$C_XD_LENGTH                    000002CE  G
SS$_ACCVIO                  =   0000000C                  UCB$C_XD_QUEUES                 =  00000006
SS$_BADPARAM                =   00000014                  UCB$C_XD_SETPRM                 =  0000000F
SS$_BUFFEROVF               =   00000601                  UCB$C_XD_VECS                   =  00000058
SS$_CTRLERR                 =   00000054                  UCB$L_CRB                       =  00000024
SS$_DEVACTIVE               =   000002C4                  UCB$L_DEVCHAR                   =  00000038
SS$_DEVICEFULL              =   00000850                  UCB$L_DEVDEPEND                 =  00000044
SS$_DEVINACT                =   000020D4                  UCB$L_DUETIM                    =  0000006C
SS$_DEVOFFLINE              =   00000084                  UCB$L_FPC                       =  0000000C
SS$_ENDOFFILE               =   00000870                  UCB$L_SVAPTE                    =  00000078
SS$_INSFARG                 =   00000114                  UCB$L_XD_CDB_VEC                   00000218  G
SS$_INSFMAPREG              =   00000344                  UCB$L_XD_HALT                      0000029C  G
SS$_INSFMEM                 =   00000124                  UCB$L_XD_JNLBUF                    00000090  G
SS$_NORMAL                  =   00000001                  UCB$L_XD_PID                       000002A0  G
STARTIO                         00000B65   R    03        UCB$L_XD_PID_VEC                   00000158  G
STARTUP                         00000BC9   R    03        UCB$L_XD_RCV_MAP                   000000C4  G
START_CONT                      00000D63   R    03        UCB$L_XD_RCV_PA                    000000FC  G
START_DEV                       00000BB8   R    03        UCB$L_XD_RCV_VA                    00000114  G
START_ERR                       00001783   R    03        UCB$L_XD_RUN                       00000298  G
START_ERR1                      00001789   R    03        UCB$L_XD_SEL0                      000002C6  G
START_RECEIVE                   00000F3F   R    03        UCB$L_XD_SEL4                      000002CA  G
START_TIMER                     00002471   R    03        UCB$L_XD_UV1BUF                    0000012C  G
START_TRIB                      00000E42   R    03        UCB$L_XD_VECS                      00000138  G
START_XMITS                     00001C39   R    03        UCB$L_XD_XMT_MAP                   000000E0  G
THRESH                          0000177D   R    03        UCB$L_XD_XMT_PA                    0000010C  G
TIMEOUT                         00002484   R    03        UCB$L_XD_XMT_VA                    00000124  G
TOO_BIG                         000002E2   R    03        UCB$M_INT                       =  00000002
TRIB_ACTIVE                     00000E74   R    03        UCB$M_ONLINE                    =  00000010
TRIB_CNT_BUFSIZ             =   00000038                  UCB$M_POWER                     =  00000020
TRIB_COUNTER                    00000157   R    03        UCB$M_TIM                       =  00000001
TRIB_COUNTER1                   0000015F   R    03        UCB$M_XD_DTRCLR                 =  00000010  G
TRIB_ERRS                       00000E8D   R    03        UCB$M_XD_FORK_PEND              =  00000008  G
TRIB_INACT                      000002DB   R    03        UCB$M_XD_FORK_XMT               =  00000020  G
TRIB_PARAM                      000000CA   R    03        UCB$M_XD_INITED                 =  00000001  G
TRIB_PRM_BUFSIZ             =   00000066                  UCB$M_XD_INITING                =  00000002  G
```

```
UCB$M_XD_PTP                    =  00000004  G
UCB$Q_XD_INFOUT                    000000AC  G
UCB$Q_XD_INPUTQ                    00000094  G
UCB$Q_XD_POST                      000000B4  G
UCB$Q_XD_QUEUES                    00000094  G
UCB$Q_XD_RCV_BUF                   000000BC  G
UCB$Q_XD_RCV_PND                   000000A4  G
UCB$Q_XD_XMT_REQ                   0000009C  G
UCB$V_ONLINE                    =  00000004
UCB$V_POWER                     =  00000005
UCB$V_TIM                       =  00000000
UCB$V_TIMOUT                    =  00000006
UCB$V_XD_DTRCLR                 =  00000004  G
UCB$V_XD_FORK_PEND              =  00000003  G
UCB$V_XD_FORK_XMT               =  00000005  G
UCB$V_XD_INITED                 =  00000000  G
UCB$V_XD_INITING                =  00000001  G
UCB$V_XD_PTP                    =  00000002  G
UCB$W_BCNT                      =  0000007E
UCB$W_BOFF                      =  0000007C
UCB$W_DEVBUFSIZ                 =  00000042
UCB$W_DEVSTS                    =  00000068
UCB$W_ERRCNT                    =  00000082
UCB$W_REFC                      =  0000005C
UCB$W_STS                       =  00000064
UCB$W_XD_CHAN_VEC                  000001D8  G
UCB$W_XD_DDT                       000002B0  G
UCB$W_XD_DLT                       000002B2  G
UCB$W_XD_POLLPRM                   000002AC  G
UCB$W_XD_QUOTA                     00000136  G
UCB$W_XD_RTT                       000002B4  G
UCB$W_XD_SLT                       000002AE  G
UCB$W_XD_SRT                       000002AC  G
UNIT_INIT                          0000016F  R      03
UNPACK_P2_BUF                      00002243  R      03
UPDATE_P2                          00002178  R      03
UV1_BUFFER_AREA                 =  00001800
VA$M_BYTE                       =  000001FF
VA$S_VPN                        =  00000015
VA$V_VPN                        =  00000009
VALIDATE_P2                        000020A5  R      03
VALIDATE_P2_CDB                    00002089  R      03
VALIDATE_P2_UCB                    00002095  R      03
VEC$B_DATAPATH                  =  00000013
VEC$B_NUMREG                    =  00000012
VEC$L_IDB                       =  00000008
VEC$L_UNITINIT                  =  00000018
VEC$W_MAPREG                    =  00000010
XD$DDT                             00000000  RG     03
XD_BSEL0_M_IEI                  =  00000001
XD_BSEL0_M_IEO                  =  00000010
XD_BSEL0_M_RQI                  =  00000080
XD_BSEL1_M_LOOPB                =  00000008
XD_BSEL1_M_MAINT                =  00000001
XD_BSEL1_M_MCLR                 =  00000040
XD_BSEL1_M_RUN                  =  00000080
XD_BSEL2_M_Q22                  =  00000008
XD_BSEL2_M_RDI                  =  00000010
XD_BSEL2_M_RDO                  =  00000080
XD_BSEL2_M_TYPE                 =  00000007
XD_BSEL2_V_ERR                  =  00000005
XD_BUF_JNX                         000024EE  RG     03
XD_CI_V_ESTAB                   =  00000001
XD_CI_V_HALT                    =  00000005
XD_CI_V_ISTRT                   =  00000003
XD_CI_V_KILL                    =  00000002
XD_CI_V_MAINT                   =  00000004
XD_CI_V_NOOP                    =  00000000
XD_CI_V_RDMODEM                 =  00000010
XD_CI_V_WTMODEM                 =  00000011
XD_DEF_BUFSIZ                   =  00000200
XD_END                             00002523  R      03
XD_FC_V_CANCEL                  =  00000010
XD_FC_V_CHGC                    =  00000004
XD_FC_V_CHGT                    =  00000005
XD_FC_V_HALTT                   =  0000000E
XD_FC_V_INITC                   =  00000001
XD_FC_V_INITT                   =  00000000
XD_FC_V_KILLT                   =  0000000F
XD_FC_V_NUMSYNC                 =  00000013
XD_FC_V_POLSS                   =  00000015
XD_FC_V_POLSTAT                 =  00000014
XD_FC_V_RCGERR                  =  0000000A
XD_FC_V_RCTERR                  =  00000009
XD_FC_V_RDGERR                  =  00000008
XD_FC_V_RDGSS                   =  0000000C
XD_FC_V_RDMODEM                 =  00000006
XD_FC_V_RDTERR                  =  00000007
XD_FC_V_RDTSS                   =  0000000B
XD_FC_V_RECV                    =  00000012
XD_FC_V_STOPC                   =  00000003
XD_FC_V_STOPT                   =  00000002
XD_FC_V_WTMODEM                 =  0000000D
XD_FC_V_XMIT                    =  00000011
XD_FILL_JNX                        000024CD  RG     03
XD_FUNCTABLE                       00000038  R      03
XD_I_V_CONTROL                  =  00000001
XD_I_V_MODE_DEF                 =  00000002
XD_I_V_RCV                      =  00000000
XD_I_V_XMIT                     =  00000004
XD_SEL6_M_ECOM                  =  00000100
XD_SEL6_M_LPOL                  =  00002000
XD_SEL6_M_RCTS                  =  00000040
XD_SEL6_M_RTSS                  =  00000020
XD_SEL6_M_UPOL                  =  00001000
XD_SEL6_M_WTSS                  =  00000080
XLATE                              00001EEE  R      03
XLATE_ALT                          00001EF2  R      03
XLATE_PID_CHAN                     00001F11  R      03
XM$M_CHR_CTRL                   =  00000040
XM$M_CHR_DMC                    =  00000020
XM$M_CHR_TRIB                   =  00000080
XM$M_STS_ACTIVE                 =  00000800
XM$M_STS_RUNNING                =  00002000
```

```
XM$V_CHR_CTRL                   = 00000006
XM$V_CHR_DMC                    = 00000005
XM$V_CHR_HDPLX                  = 00000002
XM$V_CHR_LOOPB                  = 00000001
XM$V_CHR_MOP                    = 00000000
XM$V_CHR_TRIB                   = 00000007
XM$V_ERR_FATAL                  = 00000010
XM$V_ERR_MAINT                  = 00000013
XM$V_ERR_START                  = 00000017
XM$V_ERR_THRESH                 = 00000015
XM$V_ERR_TRIB                   = 00000016
XM$V_STS_ACTIVE                 = 0000000B
XM$V_STS_BUFFAIL                = 0000000C
XM$V_STS_DISC                   = 0000000E
XM$V_STS_RUNNING                = 0000000D
XM$V_STS_TIMO                   = 00000009
XMT_ALT_START                     000002FB R      03
XMT_FDT                           0000024A R      03
XMT_OTHERS                        00000388 R      03
XMT_START                         000002E7 R      03
XMT_UV1                           0000035A R      03
```

## Psect synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| $ABS$ | 000002CE ( 718.) | 01 ( 1.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |
| $$$105_PROLOGUE | 00000069 ( 105.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| $$$115_DRIVER | 00002523 ( 9507.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG |

## Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 36 | 00:00:00.05 | 00:00:01.58 |
| Command processing | 131 | 00:00:00.50 | 00:00:03.36 |
| Pass 1 | 1655 | 00:00:47.63 | 00:03:09.77 |
| Symbol table sort | 0 | 00:00:04.50 | 00:00:18.32 |
| Pass 2 | 470 | 00:00:12.28 | 00:00:47.09 |
| Symbol table output | 2 | 00:00:00.39 | 00:00:01.75 |
| Psect synopsis output | 2 | 00:00:00.02 | 00:00:00.02 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 2299 | 00:01:05.37 | 00:04:21.90 |

```
The working set limit was 3450 pages.
437790 bytes (856 pages) of virtual memory were used to buffer the intermediate code.
There were 230 pages of symbol table space allocated to hold 3844 non-local and 564 local symbols.
6387 source lines were read in Pass 1, producing 50 object records in Pass 2.
83 pages of virtual memory were used to define 74 macros.
```

+----------------------------+
! Macro library statistics !
+----------------------------+

| Macro library name | Macros de'ined |
|---|---|
| _$255$DUA28:[SHRLIB]NMALIBRY.MLB;1 | 1 |
| _$255$DUA28:[SYS.OBJ]LIB.MLB;1 | 35 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 12 |
| TOTALS (all libraries) | 48 |

3853 GETS were required to define 48 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:XDDRIVER/OBJ=OBJ$:XDDRIVER MSRC$:XDDRIVER/UPDATE=(ENH$:XDDRIVER)+EXECML$/LIB+SHRLIB$:NMALIBRY/LIB

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XEDRIVER
LIS

XDDRIVER
LIS